碩士學位論文

VoIP 통화 품질 개선을 위한 적응 재생 버퍼 제어 기법

濟州大學校 大學院

通信工學科

姜 眞 娥

2004年 12月

VoIP 통화 품질 개선을 위한
적응 재생 버퍼 제어 기법

指導敎授　林 載 允

姜 眞 娥

이 論文을 工學 碩士學位 論文으로 提出함

2004年　12月

姜眞娥의 工學 碩士學位 論文을 認准함

審査委員長　金 興 洙　印

委　　員　左 政 祐　印

委　　員　林 載 允　印

濟州大學校 大學院

2004年　12月

Adaptive Playout Buffer Control Method for Improvement of VoIP Speech Quality

Jin-ah Kang

(Supervised by professor Jea-yun Lim)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Engineering

2004. 12.

This thesis has been examined and approved.

Thesis director, Heung-Soo Kim, Prof. of Telecom. Eng.

Thesis director, Jeong-Woo Jwa, Prof. of Telecom. Eng.

Thesis director, Jea-Yun Lim, Prof. of Telecom. Eng.

(Name and signature)

2004. 12. 20

Date

DEPARTMENT OF TELECOMMUNICATION ENGINEERING
GRADUATE SCHOOL
CHEJU NATIONAL UNIVERSITY

목 차

Abstract

In a VoIP system which support the realtime audio service, speech quality is deteriorated by the delay, the jitter, the loss, and the reversed packet order. In this thesis, APBC for receiver site is proposed, which compensate the jitter by the adaptive playout algorithm and conceal the packet loss, and align the packet order. Also, a VoIP application system is implemented, and the performance of APBC is verified on the implemented system by measuring the processing speed and the speech quality.

In order to apply 'α-adaptive' algorithm and 'jitter control procedure' as the adaptive playout algorithm, the structure and the control method of the adaptive playout buffer is designed. Also, APBC is designed to detect the reversed packet order and the loss of packet throughout RTP header parsing, and to compensate that demage based on playout buffer. APBC is implemented with C language. By the result of simulation on PC using trace data, it is confirmed that implemented APBC act well as designed; detect either silence period or talkspurt, and adaptively estimate the playout delay of a talkspurt, and write packet data received on the corresponding position of playout buffer. Also, it is found out that packet data in playout buffer is obtain by APBC read operation with the estimated playout delay. Embedded VoIP system is designed and implemented for applying APBC. And then, call test of IP Phone-to-IP Phone and IP Phone-to-Phone is performed. Finally, the performances of APBC are measured by implemented system. From the result, processing speed is 257 μ sec, which is fast enough to deal with packet being received in realtime. Also, the speech quality by MOS is improved as 18 percent compared with algorithm of fixed playout delay.

Ⅰ. 서론

음성과 데이터, 유선과 무선, 그리고 방송과 같은 서비스들은 각각의 특성에 맞게 다양한 망과 형태를 가지고 제공되고 있으나 근래에는 전 세계적인 인프라망인 인터넷을 기반으로 하나의 통합 서비스로 발전하고 있다. 이를 실현하기 위한 핵심 기술 중에 하나인 VoIP(voice over internet protocol)는 IP 상에서 음성 서비스를 지원하는 기술이며, ITU-T(international telecommunication union - tele-communication standardization sector)와 IETF(internet engineering task force) 기관을 통해 H.323 기반 시스템과 SIP(session initiation protocol) 프로토콜을 중심으로 한 표준화가 진행되고 있다. 현재 VoIP 기술의 가장 큰 쟁점은 고품질의 제공이다. 그 중 통화 품질은 NGN2003에서 발표된 인터넷 전화 사용자 불만사항에 관한 통계자료에서 33.3%를 차지하는 가장 큰 불만사항이다.(김도영과 김영선, 2004) 통화 품질은 VoIP 기술이 정착되기 위한 가장 근본적이면서도 쉽게 해결되지 않고 있는 문제로써 이는 전송망 특성에 기인한다. 즉 데이터 전송에 최적화되어 있는 IP 망을 이용하여 실시간으로 이루어지는 통화 서비스를 제공하고자 하기 때문이다. 따라서 VoIP 통화 품질 개선을 위해서는 IP 망 특성을 반영한 연구가 필요하다.

패킷 교환 방식이 사용되는 IP 망에서는 지연, 지터(jitter), 패킷 손실, 패킷 순서 뒤바뀜 현상 등이 발생하고 이는 통화 품질을 저해한다. 긴 지연은 실시간으로 이루어지는 대화를 불편하게 하고 에코(echo) 현상을 초래할 수 있다. 지터는 IP 망이 트래픽 상태에 따라 전송 지연을 가변함으로 인해 수신단에서의 패킷 수신 간격이 불규칙해지는 현상을 말하며, 이는 재생 시에 뒤늦게 도달함에 따른 패킷 손실을 발생시킨다. 패킷 손실은 이러한 경우 외에도 IP 망 내에서 트래픽의 병목현상 등에 의해 손실되는 경우에 발생된다. 패킷 손실과 패킷 순서 뒤바뀜 현상은 음성 신호를 부정확하게 복원하여 수신단의 음성 품질을 매우 떨어뜨린다.

VoIP 시스템에서는 전송 지연을 고려하여 지터를 보상하는 방법으로는 고정

재생 지연 방식과 적응 재생 지연 방식이 있다. 고정 재생 지연 방식은 음성 세션이 시작될 시에 모든 패킷에 대한 재생 지연 크기를 고정시키는 것이다.(유승화 2002) 적응 재생 지연 방식은 음성 세션 동안에 재생 지연을 적절하게 조절하는 것으로, 최근 몇 년 동안 이러한 방식을 이용하는 적응 재생 알고리즘들이 논문을 통해 제안되고 있다. 현재까지 제안된 적응 재생 알고리즘에는 자동 회귀(AR: auto-regressive) 추정 기반 알고리즘, 적응 필터 기반 알고리즘 그리고 이전 지연들에 대한 히스토그램 기반 알고리즘 등이 있다.(Ramjee 등, 1994),(DeLeon과 Sreenan, 1999),(Moon 등, 1998),(Kansal과 Karandikar, 2001). 이 중 Kansal과 Karandikar(2001)가 제안한 α-adaptive 알고리즘의 성능이 범용적인 환경에서 우수하게 나타난다. 지금까지의 VoIP 응용 시스템들은 대부분 고정된 크기를 갖는 지터 버퍼를 이용함으로써 고정 재생 지연 방식을 사용하고 있다. 그러나 고정 재생 지연 방식은 지터를 효율적으로 처리하지 못하여 만족할만한 통화 품질을 제공하기 어렵다.

본 논문에서는 적응 재생 알고리즘을 도입하여 지터를 보상하는 수신단의 적응 재생 버퍼 제어 기법(APBC: adaptive playout buffer control)을 제안한다. 이를 위해 적응 재생 버퍼의 구조와 제어 기법을 설계하고 이에 기반하여 α-adaptive 알고리즘을 구현한다. 또한 APBC 기법이 패킷 손실 은닉 기법을 수행하고 RTP(real time protocol) 헤더 정보를 이용하여 패킷 순서를 정렬하도록 구현한다. APBC를 구현함에 있어서는 C 언어를 사용한다. 구현된 APBC의 기능 검증은 PC(personal computer) 상에서 트래이스(trace) 데이터를 이용한 모의실험을 통해 수행한다. 이후에는 실제 고성능 내장형(embedded) VoIP 시스템을 구현하고 APBC 기법을 적용하여 제안 기법의 효용성을 검증한다. 이를 위해 내장형 VoIP 시스템의 하드웨어와 펌웨어를 설계 및 구현하여 IP Phone-to-IP Phone, IP Phone-to-Phone 통화 시험을 수행함으로써 기능 검증을 한다. 그리고나서는 APBC를 적용하여 알고리즘 수행 속도와 통화 품질을 측정한다. 통화 품질은 MOS(mean opinion score)로 측정하며, 그 결과는 고정 재생 지연 방식을 사용하는 방식과 비교·검토한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 배경이론이 되는 RTP 음성 패킷 구조와 VoIP 통화 품질 관련 요소, 그리고 적응 재생 알고리즘에 대해 검토한다.

Ⅲ장에서는 APBC를 설계하고 구현한 후 모의실험을 통해 기능 검증을 수행한다. Ⅳ장에서는 내장형 VoIP 시스템을 설계 및 구현하여 통화 시험을 거친 후, APBC 기법을 적용하여 성능 검증을 수행한다. Ⅴ장에서는 연구에 대한 결론을 맺는다.

Ⅱ. VoIP 통화 품질과 적응 재생 메커니즘

VoIP 시스템에서 송신단은 주기적으로 생성되는 음성 프레임 데이터를 RTP 프로토콜로 인코딩하여 패킷망을 통해 전송한다. 전송되는 패킷은 IP 망에서 지연과 지터, 손실, 그리고 순서 뒤바뀜 현상을 겪게 되고, 이에 따라 통화 품질은 크게 떨어진다. 그 중 지터를 보상하기 위해 수신단에서는 수신되는 음성 프레임 데이터들을 재생 지연만큼 일정하게 지연시킨 후에 재생하는데, 이러한 재생 지연 길이를 지연-손실 간의 상반 관계에 기반하여 적응적으로 추정하는 적응 재생 알고리즘들이 제안된다.

본 장에서는 VoIP 통화 품질과 적응 재생과의 관계를 이해하기 위해 RTP 기반 음성 패킷 구조와 지연, 지터 개념을 설명하고, 적응 재생 메커니즘과 현재까지 제안된 적응 재생 알고리즘들을 검토한다.

1. RTP 기반 음성 패킷 구조

RTP는 대화형의 음성 또는 영상 서비스와 같이, 실시간 특성을 갖는 데이터를 종단간 전송하기 위한 인터넷 표준 프로토콜이다. 이는 페이로드(payload) 유형의 식별, 순서 번호 지정, 타임스탬프(timestamp) 표시, 그리고 전송 모니터링 기능을 제공하며, 일반적으로 응용단에서는 UDP(user datagram protocol) 상위에서 RTP를 동작시킨다. VoIP 시스템은 음성 데이터를 RTP로 패킷화하여 전송하며 음성 패킷 구조는 Fig. 1과 같다.(http://www.cs.columbia.edu/~hgs/rtp/news.html)

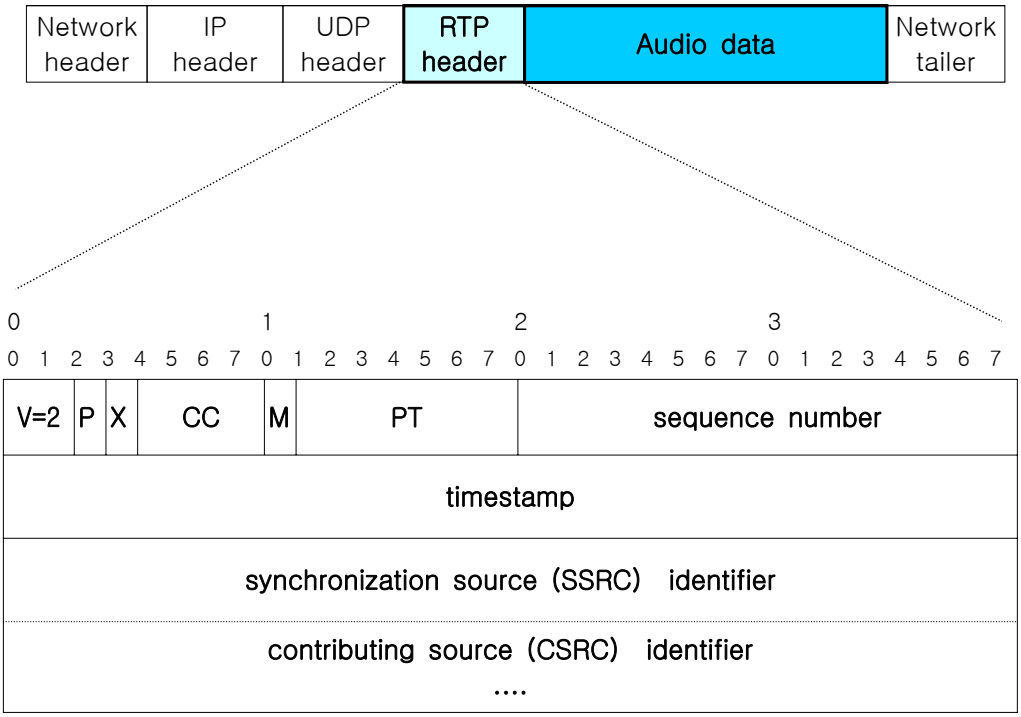


Fig. 9. VoIP audio packet structure based on RTP

RTP 헤더 크기는 12바이트이며 각 세부 사항은 다음과 같다.

- 'V=2'는 RTP 버전 2를 나타낸다.
- 'P(padding)=1'이면 패킷 크기보다 실제 데이터가 작음을 의미한다.
- 'X(extension)=1'이면 하나의 추가적인 가변 길이 헤더 확장이 있음을 의미한다.
- 'CC(CSRC count)'는 헤더 뒤에 오는 'CSRC(contributing source) identifier'의 개수를 나타낸다.
- 'M(marker)'은 패킷 스트림에서 프레임 경계와 같은 사건 발생을 나타낸다. 따라서 VoIP 송신단에서 'M' 비트에 음성 구간의 시작과 끝에 관한 정보를 실어 보냄으로써 수신단에서는 음성 구간의 끝을 판별할 수 있다.
- 'PT(payload type)'는 음성과 같은 미디어의 인코딩 유형을 나타낸다.
- 'sequence number'는 RTP 패킷이 전송될 때마다 증가된다. 따라서 수신단은 패킷 손실을 판별할 수 있다.

- ‘timestamp’는 첫 번째 옥텟(octet)의 샘플링 순간을 나타내며 동기와 지터 측정을 위해 사용된다.
- ‘SSRC(synchronization source)’는 송신자를 나타내며 같은 세션 내에서 두 개의 송신자가 같은 ‘SSRC identifier’를 갖지 않는다.
- ‘CSRC(contributing source) identifier’ 목록은 패킷 내의 페이로드를 송신한 송신자를 나타낸다. 만일 한 개의 ‘SSRC’만 존재한다면 ‘CSRC’는 0으로 표시된다. CSRC의 개수는 CC 필드에 나타난다.

(유승화, 2002), (http://www.ietf.org/rfc/rfc1889.txt)

2. VoIP 망에서의 통화 품질 요소

VoIP 망에서 다음의 세 가지 요소들이 실질적으로 음성 품질에 영향을 준다. 그것은 명확성, 종단간 지연, 그리고 에코이다. 명확성은 음성 신호의 충실성, 명확, 소리 왜곡의 해소 정도, 인지 가능성을 말하고, 종단간 지연은 음성 신호가 송신자로부터 수신자에게 전달될 때까지의 시간을 말하며, 에코는 송신자의 음성이 송신자의 귀에 되돌아오는 현상을 말한다. 본 절에서는 이 중 통화 품질에 영향을 주는 종단간 지연과 지터에 대해 설명한다.(유승화, 2002)

1) 지연

공중전화망에서는 망 교환 지연이 상대적으로 작아서, 지연은 전송 거리에 영향을 받는다고 볼 수 있다. 즉 공중전화망에서의 지연은 장거리 전송에서 발생할 수 있으나, 그 또한 통화 품질에 문제가 되지 않는 정도이다. 그러나 IP망은 패킷 교환 방식을 사용하기 때문에 지연이 쉽게 발생하며 이는 통화 품질에 문제가 된다.

음성 통신에서 종단간 지연은 송신자의 입에서 발성된 음성이 수신자의 귀에 청취될 때까지 소요되는 시간으로 정의되며, VoIP 망에서 발생하는 지연은 Fig. 2에 나타낸 것과 같이 IP 망 지연과 VoIP 장비 지연으로 구분된다.

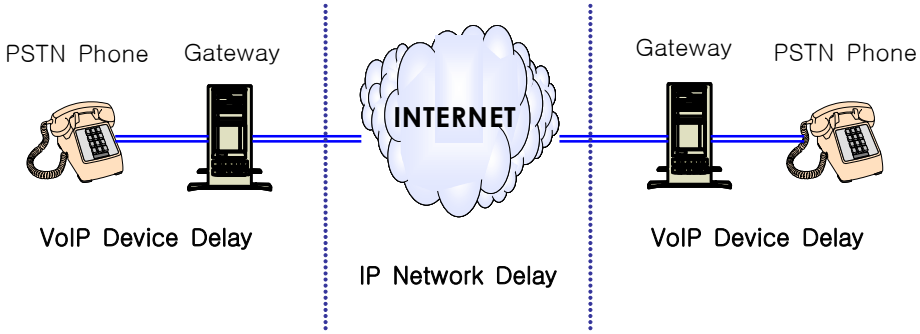


Fig. 10. End-to-end delay in VoIP network

IP 망에서 발생하는 지연은 전달 지연, 직렬화 지연, 처리 지연의 세 가지로 분류된다. 전달 지연은 패킷이 광섬유나 구리선을 매체로 하는 망을 통해 광속으로 이동하는데 걸리는 시간으로 사람의 청각으로는 인식되지 않는다. 직렬화 지연은 음성 정보의 비트나 바이트를 인터페이스에 정확하게 위치시키는데 걸리는 시간이며 이는 상대적으로 작기 때문에 문제가 되지 않는다. 그러나 패킷 처리 지연과 패킷 스위칭·라우팅 지연 그리고 대기열 지연을 포함하는 처리 지연은 실시간 통신상에 문제를 일으킨다.

패킷 처리 지연은 라우터를 통해 패킷을 받고, 처리하고, 전달하는데 소요되는 시간으로 패킷의 크기와 전송 속도에 의존한다. 패킷 스위칭·라우팅 지연은 패킷 헤더를 분석하고 라우팅 테이블을 찾아서 패킷을 출력 포트로 보내는데 소요되는 시간으로 라우터의 구조 및 라우팅 테이블의 크기에 의해 결정된다. 대기열 지연은 망 내에서 패킷들의 불규칙한 도착 현상과 전송 속도의 제한으로 인해 스위치와 라우터의 입·출력 포트에서 발생하는 지연을 말한다. 이는 포트당 패킷 크기와 트래픽, 통계적 분산, 처리 속도에 의해 결정된다.

VoIP 게이트웨이(gateway) 또는 VoIP 단말기에서 발생하는 장비 지연에는 아날로그 음성을 디지털 음성으로 인코딩하고, 반대로 디지털 음성을 아날로그 음성 신호로 디코딩하는 데 소요되는 코덱 지연이 있다. 따라서 압축 과정이 복잡할수록 더 많은 지연이 초래된다. 또한 송신단에서는 음성 정보를 패킷화할 때에 패킷화 지연이 발생된다.

Fig. 3은 종단간 지연에 따라 사람이 느끼는 통화 품질을 나타낸 것이다.

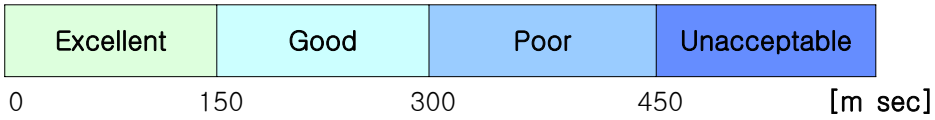


Fig. 11. Speech quality by end-to-end delay

2) 지터

VoIP 망에서의 종단간 지연에 가장 큰 영향을 미치는 요소는 대기열 지연이다. 이러한 망 내의 지연 요소가 변함으로 인해 패킷들의 도착 시간 간격이 불규칙해지는 현상을 지터라고 한다. 이러한 지터를 무시하여 음성 패킷이 도착되는 대로 재생하면 쉽게 인식하기 어려운 소리로 복원되어 음성 품질은 매우 저하된다. 따라서 수신단에서는 지터를 보상하여 원래의 규칙적인 음성 프레임 간격대로 재생해야 하며 이는 다음과 같은 방법으로 수행된다.

수신 패킷의 RTP 헤더를 분석하여 순서 번호와 타임스탬프 값을 얻고, 이를 이용하여 현재 패킷에 대한 망 지연을 측정한다. 매 수신 패킷에 대해 측정된 망 지연 데이터들에 기반하여 종단간 재생 지연을 결정하며 그만큼 패킷을 지연시킨 후에 재생한다. 일반적으로 이러한 방법에는 고정 재생 지연 방식과 적응 재생 지연 방식이 있다. 고정 재생 지연 방식인 경우에 만일 송신자가 t 시간에 음성 패킷을 생성하여 q 시간만큼 지연시킨 후에 재생을 시도한다고 가정하면, 수신자는 $t+q$ 시간에 음성 패킷을 재생하게 된다. 이 때 음성 패킷이 계획된 시간 이전에 도착하지 못하면 패킷을 폐기하여 손실시킨다.

3. 적응 재생 지연 조절

Fig. 4에서 좌측의 계단 그래프는 송신단에서 규칙적인 시간 간격으로 음성 패킷을 생성하고 있음을 나타낸다. 이에 대해 우측의 계단 그래프는 이를 불규칙한

시간 간격으로 수신하고 있음을 보여주고 있다. 그리고 수신단에 그어진 두 점선은 수신 패킷에 대한 재생 시간을 표시한 것으로써, 첫 번째 패킷이 수신된 이후 어느 정도 시간이 지난 후부터 규칙적인 시간 간격으로 재생되고 있음을 알 수 있다.

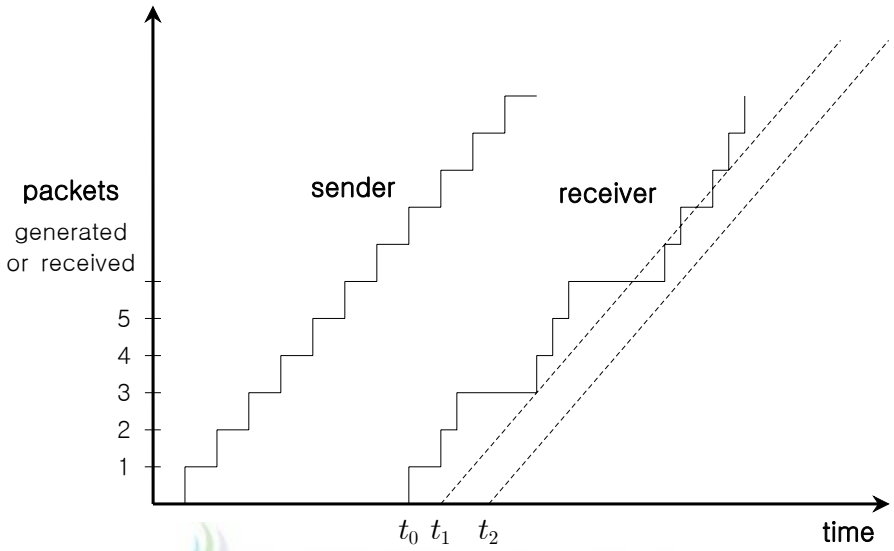


Fig. 12. Generation and reconstruction of packetized audio with fixed playout delay

이와 같이 지터가 제거되어 규칙적인 시간 간격으로 손실 없는 재생을 하기 위해서는 가장 큰 지연을 겪는 패킷이 예정된 재생 시간 이내에 수신될 만큼 패킷들을 저장(버퍼링)하는 것과 관계된다. 이는 Fig. 4에서 t_2 시간에 재생을 시작하는 것과 같다. 이 방법으로는 단지 하나의 패킷만이 과도하게 지연되는 경우에도 매우 큰 재생 지연 t_2 를 선택하게 되며 이는 어느 정도의 패킷 손실이 허용됨에도 불구하고 모든 패킷들을 과도하게 지연시킨다. 대신에 t_1 시간에 재생을 시작한다면 종단간 지연은 줄지만 패킷 손실이 발생된다. 따라서 보다 작은 t_1을 선택하면서 패킷 손실 허용치를 만족시킬 수 있는 알고리즘이 필요하다.

음성 세션 시작 시에 재생 지연을 설정하는 고정 재생 지연 방식은 만족스러운 음질을 얻지 못한다. 따라서 재생 알고리즘들은 음성 스트림 동안에 망 지연이 변함에 따라 적응적으로 재생 지연 t_1을 조절하고 이에 따른 패킷 손실이 비교적 낮

게 유지되도록 요구된다. 일반적으로 이러한 조절 과정은 음성 스트림 내의 묵음 주기에서 수행되는데, 이는 지연 조절로 인해 발생될 수 있는 묵음 구간의 확장 또는 압축이 통화 품질을 크게 저해하지 않는다는 가정을 토대로 한다.(Ramjee 등 1994)

1) 트래이스에 대한 평균 재생 지연과 패킷 손실의 측정

여기에서 말하는 트래이스는 하나의 음성 세션에 대한 패킷들의 정보가 기록된 데이터 집합을 말한다. 이 때의 패킷 정보는 음성 구간의 발생 회수, 각 음성 구간에 수신된 패킷 수, 각 패킷에 대한 송신단 패킷 생성 타임스탬프와 수신단 패킷 도착 타임스탬프, 그리고 패킷 재생 타임스탬프 등이 된다.

재생 지연은 송신단에서의 음성 패킷 생성 시간과 수신단에서의 음성 패킷 재생 시간 간의 차이로 정의된다. 음성 패킷에 대한 시간 정보를 보이고 평균 재생 지연을 명확하게 정의하기 위해 아래의 Fig. 5를 참고한다.

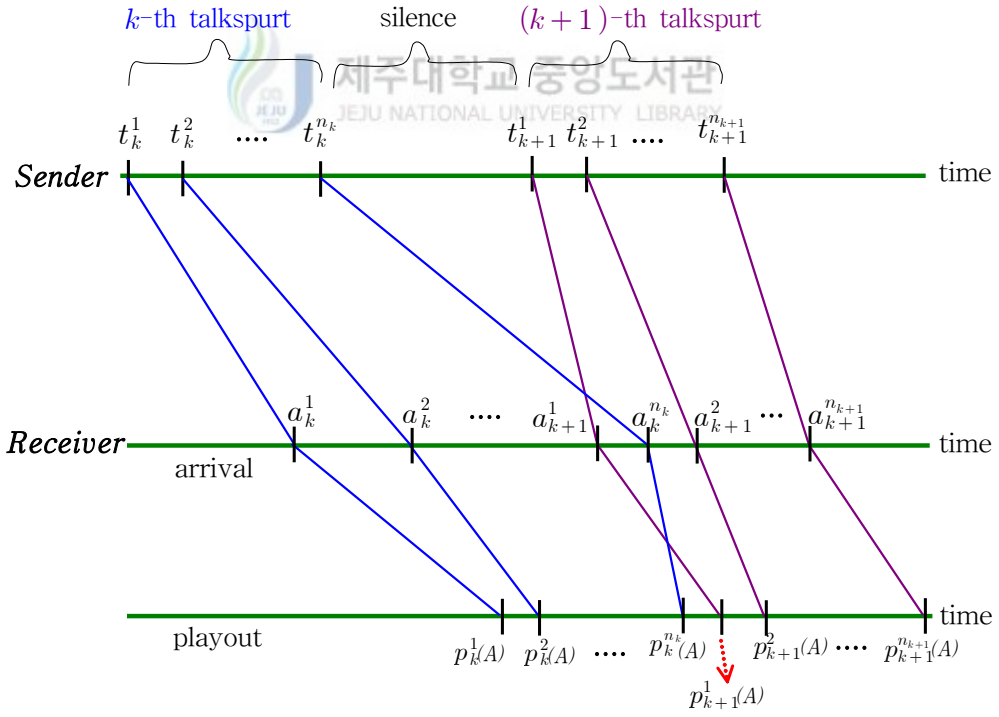


Fig. 5. Timing associated with i-th packet in the k-th talkspurt

다음은 M 개의 음성 구간으로 이루어진 트래이스에 대한 정의이다.

- t_k^i : k-번째 음성 구간의 i-번째 패킷에 대한 송신단 타임스탬프
- a_k^i : k-번째 음성 구간의 i-번째 패킷에 대한 수신단 타임스탬프
- n_k : k-번째 음성 구간내의 패킷 수이며, 수신단에서 실제로 수신된 패킷만을 고려한다.
- N : 트래이스 내의 총 패킷 수이며, 즉 $N = \sum_{k=1}^{M} n_k$ 이 된다.

패킷의 재생 시간은 수신단에서 사용되는 재생 지연 추정 알고리즘에 의존한다. 따라서 재생 알고리즘을 A라고 한다면 $p_k^i(A)$는 A 알고리즘 하에서, k-번째 음성 구간의 i-번째 패킷에 대해 추정된 재생 타임스탬프라고 한다. k-번째 음성 구간의 i-번째 패킷이 $p_k^i(A)$보다 늦게 도착하는 경우 즉, $p_k^i(A) < a_k^i$일 때에는 패킷 손실로 처리한다. 이 외의 경우에는 $(p_k^i(A) - t_k^i)$의 재생 지연으로 재생한다. k-번째 음성 구간의 i-번째 패킷이 재생 시간 이전에 도착하였는지를 나타내기 위한 지시 변수를 $r_k^i(A)$로 두면, 이는 식 (1)과 같다.

$$r_k^i(A) = \begin{cases} 0, & \text{if } p_k^i(A) < a_k^i \\ 1, & otherwise \end{cases} \quad (1)$$

알고리즘 A에 의해 재생된 총 패킷 수는 $N(A)$로 표기하며 $r_k^i(A)$를 이용하여 식 (2)와 같이 계산된다.

$$N(A) = \sum_{k=1}^{M} \sum_{i=1}^{n_k} r_k^i(A) \quad (2)$$

따라서 평균 재생 지연은 식 (3)으로 구해진다.

$$\hat{p}(A) = \frac{1}{N(A)} \sum_{k=1}^{M} \sum_{i=1}^{n_k} r_k^i(A)\,(p_k^i(A) - t_k^i) \quad (3)$$

하나의 트래이스에 N개의 패킷이 존재하고, $N(A)$개의 패킷이 재생되었다면, 손실률 l은 다음의 식 (4)와 같이 계산된다.

$$l = \frac{N - N(A)}{N} \times 100 \quad (4)$$

4. 적응 재생 알고리즘

현재까지 제안된 적응 재생 알고리즘에는 자동 회귀 추정 기반 알고리즘, 적응 필터 기반 알고리즘 그리고 이전 지연들에 대한 히스토그램 기반 알고리즘 등이 있다. Ramjee 등(1994)에서 보고된 자동 회귀 추정 기반 알고리즘들의 성능 비교를 보면 그 중 어떠한 알고리즘도 월등히 우수하다고 할 수 없다. DeLeon과 Sreenan(1999)에서 제안된 NLMS(normalized least mean squares) 필터 기반 알고리즘은 매 패킷마다 종단간 재생 지연을 갱신하는 방식을 사용하는데 이러한 방식은 재생 상의 지터 문제를 해결하기 어렵다. 실제 이 논문에서 지연-손실 성능을 시뮬레이션한 결과 또한 이전 알고리즘의 성능보다 개선되지 않았다. 이를 Kansal과 Karandikar(2001)가 매 음성 구간에 대해 재생 지연을 조절하는 방식으로 바꾸어 시뮬레이션 한 결과에서도 성능은 더욱 나빠짐을 확인하였다. Moon 등(1998)에서는 과거 지연들에 대한 히스토그램에 기반하는 알고리즘을 제안하고 기존 알고리즘들과의 성능 비교 시뮬레이션을 수행하였는데, 2% 미만의 낮은 손실 영역에서 알고리즘들에 의해 얻어지는 재생 지연이 최소 경계치의 값과 격차가 많이 벌어짐을 확인할 수 있다. 이에 따라 낮은 손실 영역에서의 지연-손실 성능을 개선하고자 한 것이 α-adaptive 알고리즘이다.(Kansal과 Karandikar, 2001)

1) 자동 회귀 추정 기반 알고리즘

자동 회귀를 이용한 평균 망 지연 d_i는 식 (5)와 같이 주어진다.

$$d_i = \alpha d_{i-1} + (1-\alpha) n_i \quad (5)$$

여기에서 n_i는 현재 수신 패킷에 대한 망 지연이고 α는 알고리즘의 수렴율을 조절하는 가중 인자이다. d_i에 대한 편차는 식 (6)과 같다.

$$v_i = \alpha v_{i-1} + (1-\alpha)\left| d_i - n_i \right| \quad (6)$$

식 (5)와 식 (6)에 의한 평균 망 지연과 이에 대한 편차는 다음 패킷에 대한 종단간 지연을 예측하는데 사용되며 이는 식 (7)과 같다.

$$ted = d_i + \beta v_i \quad (7)$$

여기에서 β는 보다 안정적으로 종단간 지연의 예측치가 실제치보다 커지도록 하는 가중 인자로 β 값이 클수록 늦게 도착함에 따른 패킷 손실량은 작아진다. 앞서 언급된 논문들에서 β값은 4로 제시되었다. 이 알고리즘에서 종단간 지연 ted의 조절은 새로운 음성 구간이 시작될 때에 하는 것이 합리적이다.

2) α-adaptive 알고리즘

식 (5)에서 α의 선택은 평균 망 지연 d_i의 AR 추정 계산에서 가장 최근의 데이터에 대한 가중에 영향을 준다. 만약 과거 측정치 개수가 1/α보다 크다면 지연 추정치는 과거에 측정한 망 지연값보다 최근에 측정한 값에 더 비중을 두게 될 것이다. 이는 TCP(transmission control protocol)의 RTT(round-trip time)를 예측하는 방법과 비슷하다. 즉 α는 d_i가 얼마나 빨리 망 지연에 대한 변화를 쫓을 것인가를 결정하며 α값의 작은 변화는 지연-손실 상반 관계에 확연한 영향을 미친다. 일부의 패킷들이 도착하고 난 뒤에 하나의 음성 구간에 대한 재생 지연을 최적의 값으로 결정할 수 있도록 α값을 적응적으로 조절하고자 한 것이 α-adaptive 알고리즘이며 이는 Fig. 6과 같이 수행된다.

1. $\alpha = 0.998002$, $increment = 0.0001$, $\alpha_2 = \alpha - increment$
2. Calculate ted based on α and ted_2 based on α_2 using (5), (6) and (7). Use ted_2 for actual playout.
3. loss1 = Loss(L, α), loss2 = Loss(L, α_2)
   /* Loss(X, Y) calculates the loss based on $\alpha = Y$ in the previous X talkspurts */
4. if loss2 < loss1
   if $\alpha_2 < \alpha$ then $\alpha_2 = \alpha - increment$
   if $\alpha_2 > \alpha$ then $\alpha_2 = \alpha + increment$
5. if loss2 > loss1
   if $\alpha_2 < \alpha$ then $\alpha_2 = \alpha - increment$
   if $\alpha_2 > \alpha$ then $\alpha_2 = \alpha + increment$
6. Repeat steps 3 to 5 every L^{th} talkspurt.

Fig. 6. Process of α-adaptive algorithm

재생 지연은 매 음성 구간마다 조절된다. 여기에서 '$increment$'는 α의 매우 작은 변화가 손실과 재생 지연의 매우 큰 변화를 유도하기 때문에 α보다 매우 작게 설정된다.

3) 지터 제어 절차

통화 중 묵음 구간 길이의 변화는 통화 품질에 상대적으로 영향을 덜 준다. 따라서 적응 재생 알고리즘들은 두 음성 구간 사이의 묵음 구간 동안에 재생 지연을 조절하며, 이는 음성 구간에서 발생되는 지터를 묵음 구간으로 밀어 넣는 것이라고 할 수 있다. 그러나 이러한 알고리즘의 수행으로 인해 묵음 구간이 거의 없어져버리는 묵음 구간 왜곡 현상이 간혹 발생된다. 이러한 묵음 구간의 왜곡으로 인해 통화 품질이 저하됨을 보상하기 위해 Kansal과 Karandikar(2001)가 제안한 지

터 제어 절차는 다음과 같다. 재생 지연의 조절은 새로운 음성 구간의 첫 번째 패킷이 도착될 때 이루어진다. 이 단계에서는 이미 이전 음성 구간의 마지막 패킷과 새로운 음성 구간의 첫 번째 패킷이 둘 다 수신된 상태이며 따라서 두 패킷의 생성 타임스탬프 간의 차를 구하여 실제 발생된 묵음 구간 길이를 측정할 수 있다. 또한 새로운 음성 구간의 첫 번째 패킷에 대해 예측된 재생 시간과 이전 음성 구간의 마지막 패킷이 재생된 시간 간의 차를 구함으로써, 재생 알고리즘에 의해 추정된 묵음 구간 길이를 측정할 수 있다. 만일 추정된 묵음 구간이 허용 범위 내에서 압축되었다면 추정치를 그대로 수용한다. 그렇지 않고 압축 허용 범위를 초과한다면 허용 범위에 근간한 최소 묵음 구간 길이로 설정한다. 이에 따라 새로운 음성 구간의 재생 지연을 재조정한다.

Ⅲ. 적응 재생 버퍼 제어 기법(APBC)의 설계 및 구현

Ⅱ장에서는 RTP 음성 패킷 구조와 지연-손실 간의 상반 관계를 감안한 적응 재생 알고리즘을 검토함으로써 수신단에 적용할 수 있는 VoIP 통화 품질 개선 알고리즘의 설계 방향을 얻었다.

본 장에서는 Ⅱ장에서 설명된 α-adaptive 알고리즘과 지터 제어 절차를 적용하여 재생 지연 결정 및 묵음 구간 보상을 수행하고, 수신되는 음성 패킷의 RTP 헤더 정보를 이용하여 패킷 정렬과 패킷 손실 보상 알고리즘을 적용할 수 있는 일련의 과정을 설계한다. 또한 각각의 알고리즘들이 유기적으로 수행될 수 있도록 APBC 구조를 설계한다. 그 후 APBC를 실제 C언어로 구현하여 모의 실험함으로써 APBC의 기능을 검증한다.

1. APBC 전체 구조

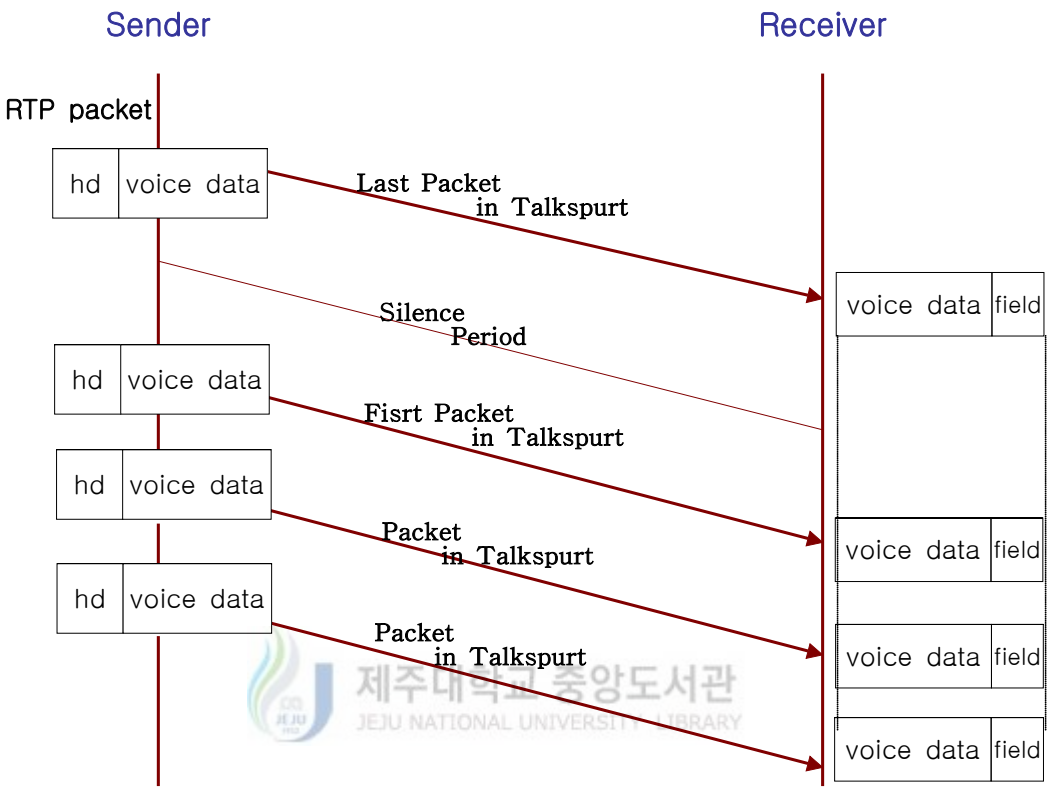
본 연구에서 적용하고자 하는 α-adaptive 알고리즘은 존재하는 적응 재생 알고리즘들의 성능이 낮은 패킷 손실 영역에서 저하됨을 인식하고 이를 개선하고자 한 것으로 종단간의 망 지연을 추정함에 있어서 AR 방법의 α값을 조절한다. 이는 다음과 같이 수행된다. 이전의 음성 구간들에 대한 트래이스에서 일정한 표본 구간을 정하고, 그 구간에 가변시킨 α값을 적용하여 패킷 손실을 구한다. 이 때 보다 낮은 패킷 손실을 유발하는 α값을 선택하고 이를 현재 음성 구간에 대한 재생 지연을 추정하는데 적용한다. α-adaptive 알고리즘의 성능 검증은 Kansal과 Karandikar(2001)에서 다양한 음성 세션 트래이스를 가지고 오프라인 상에서 이루어졌다.

APBC는 이러한 α-adaptive 알고리즘을 실제 VoIP 시스템에 적용하고 이 외에도 부가적인 통화 품질 개선 알고리즘을 수행하도록 본 연구에서 설계한 재생 버

퍼 구조와 제어 과정을 말한다. 먼저 APBC의 적응적 재생 지연 조절 기능은 다음과 같이 수행된다. APBC는 음성 세션의 시작 단계에서 재생 버퍼를 생성한다. 이후부터는 음성 패킷을 수신할 때마다 RTP 헤더를 분석하고 음성 구간과 묵음 구간의 주기를 판별하며 그에 따른 α-adaptive 알고리즘을 수행한다. 즉 묵음 구간이 끝나고 새로운 음성 구간이 시작될 시에는 재생 지연을 추정하여 재생 버퍼에서의 버퍼링 지연을 조절하며 결정된 버퍼링 지연값을 프레임 수로 변환시킨다. 변환 과정에서 코덱 지연과 추정된 재생 시간 사이에는 약간의 오차가 발생할 수 있으며 이 때 오차가 작아지는 쪽으로 추가적인 버퍼링 지연을 증가시킨다. 동일한 음성 구간 동안에는 음성 구간이 시작될 시에 결정된 버퍼링 프레임 수를 유지하도록 수신 패킷들을 버퍼 내에 위치시킨다. 또한 α값을 결정하는데에 필요한 이전 패킷들에 대한 트래이스를 음성 세션 동안 계속하여 기록하고 갱신한다.

이 외에도 APBC는 분석된 RTP 헤더의 타임스탬프 값과 순서 번호를 동시에 비교하여 정확한 패킷 생성 순서를 판단하고 이에 따라 패킷을 순차적으로 정렬한다. 또한 망에서의 패킷 손실 발생 여부를 검사하여 손실된 패킷의 재생 시점에서 이전에 가장 근접하게 수신된 패킷을 반복 재생해줌으로써 패킷 손실을 은닉한다.

아래의 Fig. 7은 IP 망을 이용한 음성 통신 상에서의 APBC 동작을 나타낸다. 기본적으로 음성 통신은 양방향으로 이루어지나 설명을 위해 송신자와 수신자으로 구분하여 표시하였다. 그림에서 보이는 바와 같이 송신단은 음성 코덱에 의해 주기적으로 생성되는 음성 프레임 데이터를 RTP 패킷화하여 전송한다. 또한 묵음 구간 동안에는 패킷을 전송하지 않는다. 이에 대해 수신단에서는 수신 패킷들을 APBC 처리하여 재생 버퍼에 저장하며, 하나의 음성 구간에 대한 재생 지연의 추정과 지터 제어 과정을 수행하는 시점은 이전 음성 구간의 종료가 검출될 때가 된다. 이러한 APBC의 과정을 순서도로 표시하면 Fig. 8과 같다.



Transfer packet

<info>
- generated timestamp by transfer system (on timestamp field of RTP header)
- sequence number by protocol (on corresponding field of RTP header)

<voice data> on RTP data field

Received packet

<info>
- generated timestamp given by RTP header
- arrived timestamp measured by receiver system
- playout delay(playout time) determined by receiver algorithm

<voice data> given by RTP data

Fig. 7. Voice communication between a sender and a receiver

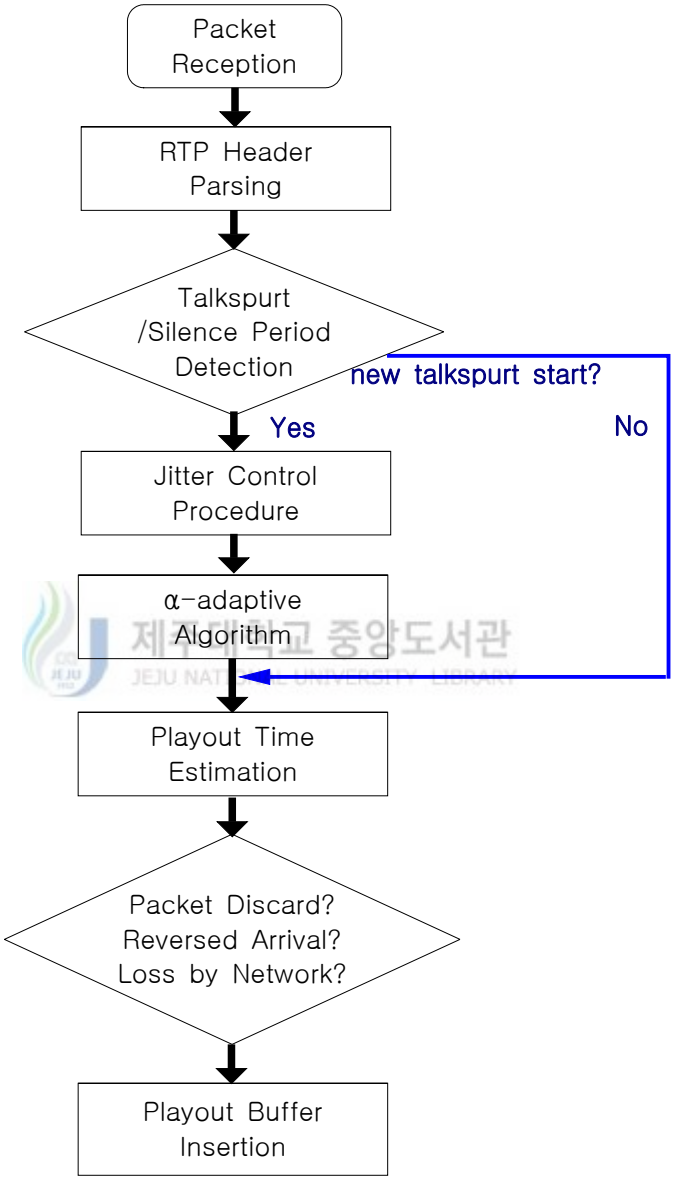


Fig. 8. APBC flowchart

2. APBC 세부 구조

1) 음성 구간과 묵음 주기의 결정

음성 활동은 음성 구간과 묵음 구간의 두 상태가 교대하는 것으로 모델링 된다.(Pinto와 Christensen, 1999) 이전에는 묵음 구간일 때에 묵음 패킷을 전송하는 방식을 사용하였으나 근래에는 대역폭을 절약하기 위해 음성 구간일 때의 음성 패킷만을 전송하도록 설계되고 있다. 따라서 본 연구에서도 묵음 구간 동안에는 패킷을 전송하지 않는 송신단을 고려하여 APBC 수신단에서 음성 구간과 묵음 구간을 식별할 수 있도록 구현한다. 이는 RTP 헤더 정보를 이용함으로써 이루어진다.

Fig. 9는 하나의 음성 구간에 대한 시작과 끝을 나타낸 것이다.

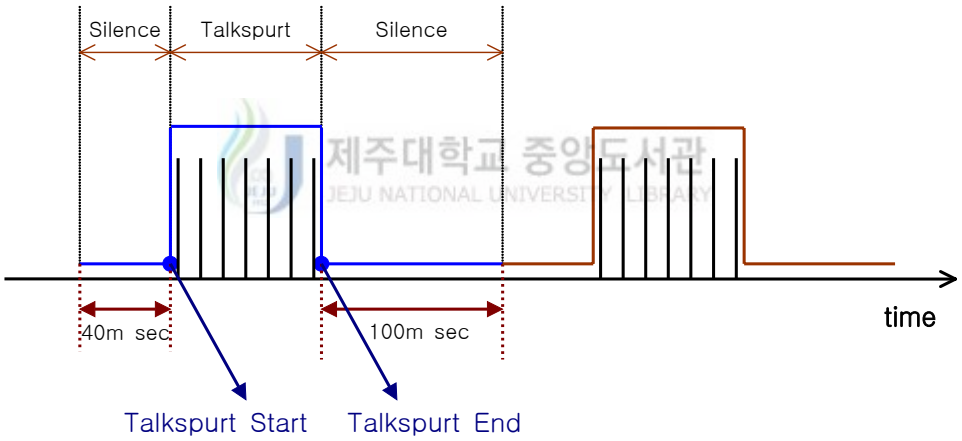


Fig. 9. Start and end of a talkspurt

매 음성 패킷 수신 시에 현재 패킷과 이전 패킷에 대한 RTP 헤더의 타임스탬프 값을 비교하여 그 차이가 단일 음성 프레임 크기 이상이 되면, 묵음 구간이 끝나고 새로운 음성 구간이 시작되었다고 판단할 수 있다. 그러나 알고리즘 수행 효율을 위하여 NeVot(network voice terminal)의 기본 설정과 같이 하나의 음성 프레임 크기보다 큰 140 [m sec]를 기준으로 하도록 한다. 즉 음성 구간이 시작되기 전에는 40 [m sec] 이상의 묵음이 유지되고 음성 구간이 끝나게 되면 100 [m

sec] 이상의 묵음이 유지된다고 가정하여 패킷간의 타임스탬프 차이가 140 [m sec] 이상이면 새로운 음성 구간이 시작되었다고 판단하도록 하고, 그렇지 않으면 동일한 음성 구간 내의 패킷이라고 판단하도록 한다.

2) APBC 재생 지연 조절 메커니즘

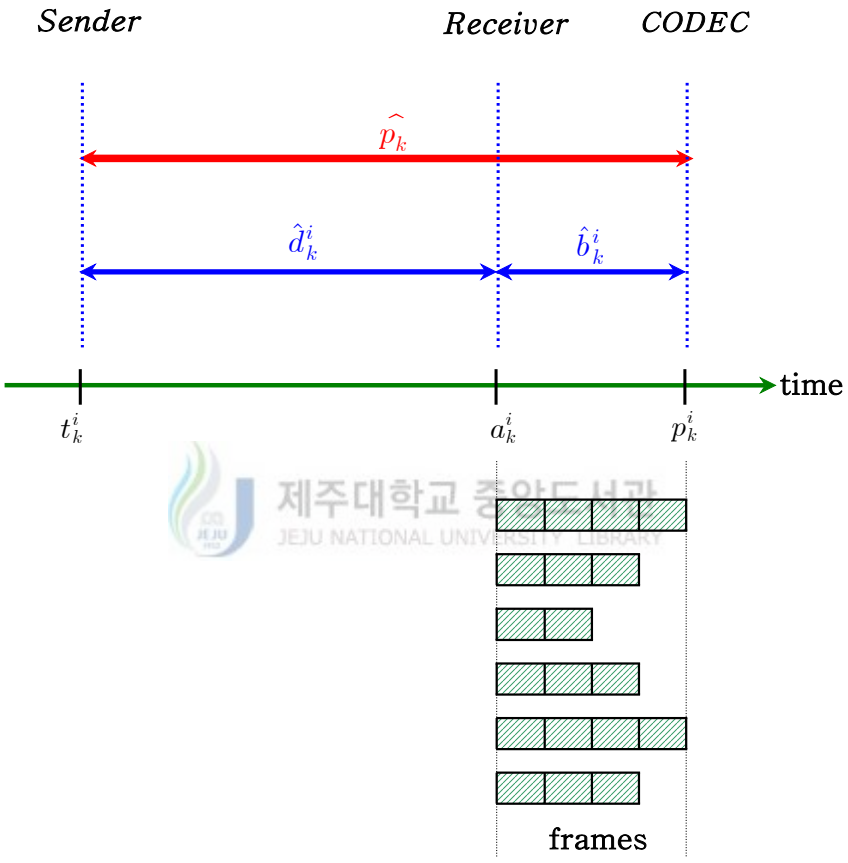


Fig. 10. Playout delay adjustment in APBC

Fig. 10을 참조하여 k-번째 음성 구간의 i-번째 패킷에 대해 다음과 같이 정의한다.

- $\hat{p}_k$: 재생 지연
- $\hat{d}_k^i$: 망 지연

- $\hat{b}_k^i$: 버퍼링 지연
- $\hat{d}_{codec}$: 코덱 지연

α-adaptive 알고리즘에 의해 k-번째 음성 구간에 대한 재생 지연을 추정한 후에 버퍼링 지연은 식 (8)과 같이 계산된다.

$$\hat{b}_k^i = \hat{p}_k - \hat{d}_k^i \tag{8}$$

즉, 각 수신 패킷에 대한 망 지연이 변하기 때문에 버퍼링 지연을 조절하여 하나의 음성 구간에 대한 재생 지연을 일정하게 유지시키도록 한다. 버퍼링 지연을 Fig. 10에서와 같이 코덱이 한번에 처리하는 단위 프레임의 개수로 변환시킨다. 즉, a_k^i 시점에 도달한 패킷이 재생 버퍼에 인가된 후 코덱에서 재생되기까지의 버퍼링 지연은 (버퍼 대기 프레임 수 × 코덱 지연)이 된다. 따라서 버퍼 대기 프레임 수를 조절하여 버퍼링 지연을 가변한다. 버퍼 대기 프레임 수는 식 (9)에서 N_k^i와 같다.

$$\frac{\hat{b}_k^i}{\hat{d}_{codec}} = \frac{p_k^i - a_k^i}{\hat{d}_{codec}} = N_k^i + \hat{r}_k^i \tag{9}$$

음성 구간의 시작에서 재생 지연을 프레임 수로 변환한 이후부터는 수신 패킷들에 대한 재생 시간을 프레임 간격으로 추정하며 이는 식 (10)와 같다.

$$p_k^i - p_k^{i-1} = t_k^i - t_k^{i-1} \tag{10}$$

3) 묵음 구간 보상

Fig. 11은 APBC에서 보상하고자 하는 묵음 구간 압축 현상을 가정하여 나타낸 것이다.

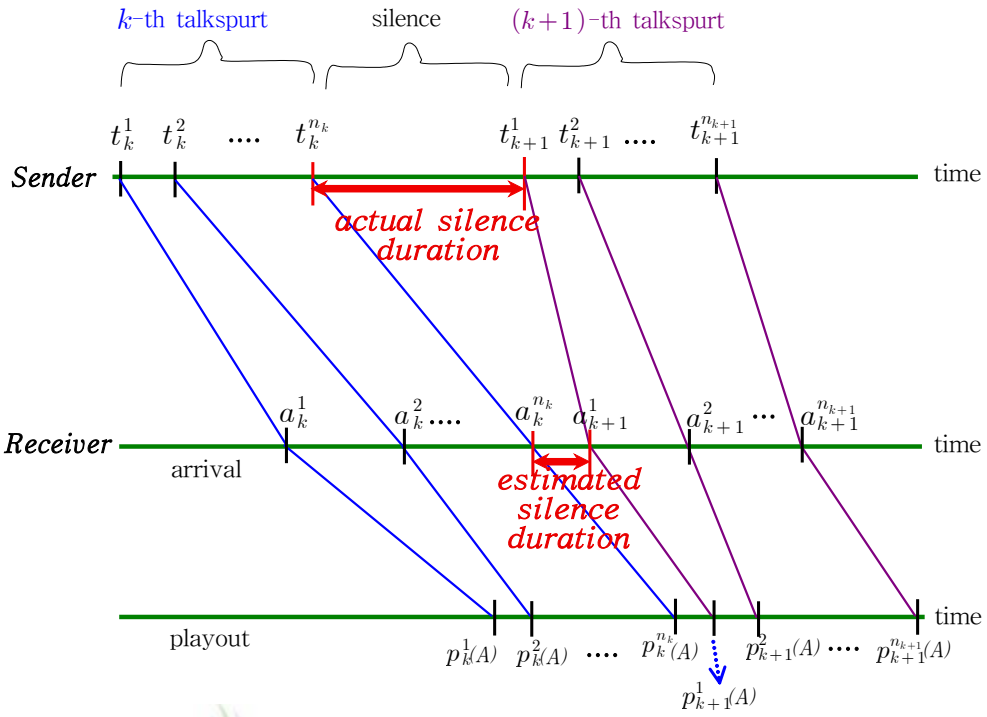


Fig. 11. Totally compressed silence duration caused by playout delay adjustment algorithm

발성자에 의해서 송신단에서 실제로 생성된 묵음 구간은 Fig. 11의 실제 묵음 구간('actual silence duration')으로 식 (11)과 같이 계산된다.

$$\hat{s}_{act\_k} = t_k^1 - t_{k-1}^{n_{k-1}} \tag{11}$$

수신단에서는 α-adaptive 알고리즘에 의해 매 음성 구간에 대한 재생 지연을 조절함에 따라 묵음 구간이 실제와 달라질 수 있으며 이는 Fig. 11의 추정 묵음 구간('estimated silence duration')과 같다. 이는 식 (12)와 같이 계산된다.

$$\begin{aligned} \hat{s}_{esti\_k} &= p_k^i - p_{k-1}^{n_{k-1}} \\ &= (t_k^1 - t_{k-1}^{n_{k-1}}) + (\hat{p}_k - \hat{p}_{k-1}) \end{aligned} \tag{12}$$

(11)과 (12) 식에 의해 묵음 구간의 왜곡은 (12) 식의 두 번째 항인 $(\hat{p}_k - \hat{p}_{k-1})$ 만큼 발생하며, $\hat{p}_k(A) > \hat{p}_{k-1}(A)$일 때에는 묵음 구간이 확장되었음을, 그 반대일 때에는 압축되었음을 알 수 있다. 만일 묵음 구간의 압축률이

허용 범위(실제 묵음 구간에 대한 백분율로 표현되는)를 초과하는 경우에는 재생 지연을 늘리도록 하는데, 이 때의 재생 지연은 식 (13)을 만족하는 $\hat{p}_k'$가 된다. 식 (13)에서 s_{toler_k}는 허용되는 최소 묵음 구간을 표기한 것이다.

$$\hat{p}_{k-1} - \hat{p}_k' = s_{toler\_k} \tag{13}$$

본 연구에서는 허용 범위를 50% 이하인 경우에 대해 구현하며 따라서 허용되는 최소 묵음 구간은 식 (14)와 같다.

$$\hat{s}_{toler\_k} = \frac{t_k^1 - t_{k-1}^{n_{k-1}}}{2} \tag{14}$$

이러한 원리에 근간하여 묵음 구간 보상을 수행하는 과정은 다음과 같다. 우선 음성 구간의 시작에서 $\hat{s}_{act_k}$와 $\hat{s}_{esti_k}$를 구하여 묵음 구간 보상 여부를 판단한다. 보상 하는 경우 즉, 허용 범위 이상으로 압축된 경우에는 $\hat{s}_{toler_k}$에 해당하는 묵음 프레임 수를 결정한 후 재생 버퍼 상에 그만큼의 간격을 두고 첫 번째 프레임 데이터를 쓴다. 보상하지 않는 경우에는 $\hat{s}_{act\_k}$ 에 해당하는 묵음 프레임 수를 결정하여 그 만큼의 간격을 두어서 첫 번째 프레임 데이터를 쓴다.

3. APBC 재생 버퍼

1) 재생 버퍼 생성 및 APBC 기록 동작

Fig. 12에서 우측 하단의 구조체는 링크드 리스트(linked list) 형태의 재생 버퍼를 구성하는 단위 프레임 구조체를 나타낸다. 이는 하나의 수신 패킷에 대한 정보 멤버(member)와 음성 데이터 멤버, 그리고 단위 프레임 구조체간의 연결 주소 멤버를 갖는다. 정보 멤버인 상태 표시자('status flag')는 음성 프레임, 망에서 손실된 프레임, 뒤바뀐 순서로 전송된 프레임, 그리고 묵음 프레임을 구분하여 나타내고 순서번호('sequence number')는 RTP 헤더에서 분석된 순서 번호를 나타낸다. 음성 데이터 멤버인 음성 데이터('voice data')는 음성 코덱이 한번에 처리할 수 있

는 단위 프레임 크기의 데이터를 저장하며, 연결 주소 멤버인 이전 프레임 포인터('previous frame pointer')와 다음 프레임 포인터('next frame pointer')는 해당 프레임 구조체를 기준으로 하여 직전과 직후 프레임 구조체의 포인터를 가리킨다. Fig. 12에 표시된 번호를 따라 이러한 단위 구조체들을 순차적으로 생성하고 링크시킴으로써 하나의 재생 버퍼를 생성한다.

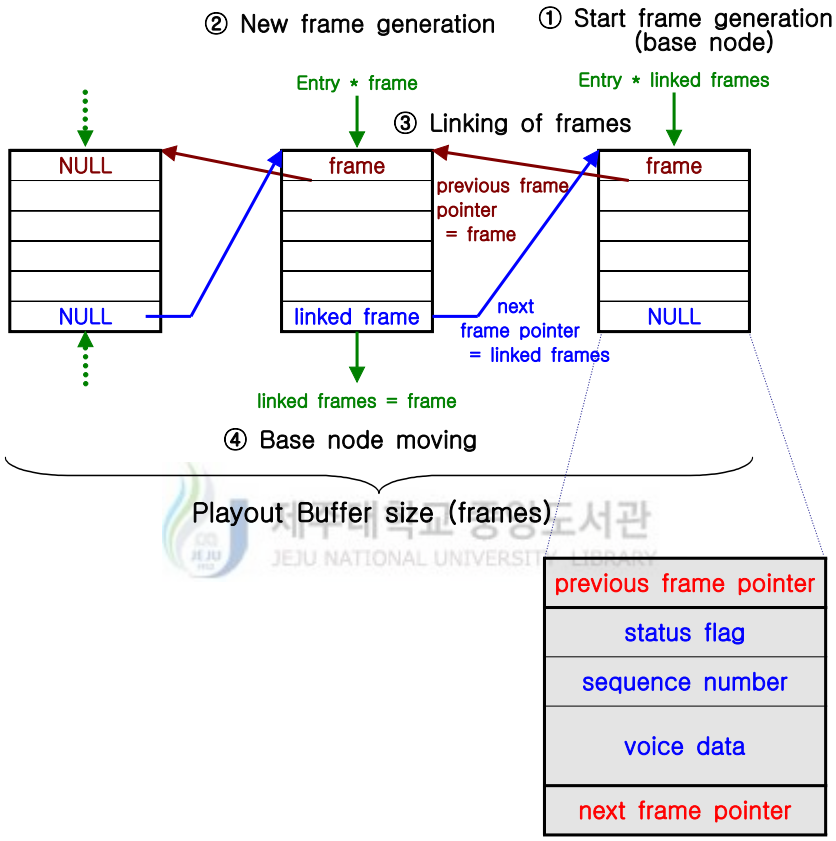


Fig. 12. Process of playout buffer generation

Fig. 12와 같은 구조를 갖는 재생 버퍼에 수신된 음성 패킷의 프레임을 기록하는 과정은 다음의 의사 코드(pseudo code)로 나타내었다.

```
/* check status flag */
packet_interval=|current_sequence_number - previous_sequence_number|;
decision_wrap_around(packet_interval);
check_status_flag();
    → status_flag = PACKET_NORMAL;
    → status_flag = PACKET_AFTER_LOSS;
    → status_flag = PACKET_REVERSED;

/* decide whether this packet to be discarded or not */
if( p_k^i > a_k^i )
    status_flag = DISCARD_PACKET;
    return;

/* write packet data to appropriate location of frame within playout buffer */
switch ( status_flag )
  case PACKET_NORMAL :
       /* write to current frame */
      current_write_frame->flag = PACKET_NORMAL;
      current_write_frame->sequence_number = current_sequence_number;
      memcpy( current_write_frame->voice_data, current_voice_data, 24 );
      current_write_frame = current_write_frame->next;
  case PACKET_REVERSED :
     /* search the appropriate location of frame */
      frame = current_write_frame->previous_frame;
      for( i = 0; i < packet_interval+1; i++ )
           frame = frame->previous_frame;
           if ( (current_sequence_number >= frame->sequence_number) &&
                (frame->status_flag == PACKET_AFTER_LOSS) )
                  /* write to current frame */
                     frame->status_flag = PACKET_NORMAL;
                     memcpy( frame->voice_data, current_voice_data, 24 );
  case PACKET_AFTER_LOSS :
     /* check status_flag of lost frames */
     for( i = 0; i < number_of_lost_frames; i++ )
          current_write_frame->flag = PACKET_LOSS;
          current_write_frame->sequence_number =
                                  previous_sequence_number + i + 1;
          current_write_frame->data = 0x00;
          current_write_frame = current_write_frame->next;
     /* write to current frame */
     current_write_frame->flag = PACKET_NORMAL;
     current_write_frame->sequence_number = current_sequence_number;
     memcpy( current_write_frame->voice_data, current_voice_data, 24 );
     current_write_frame = current_write_frame->next;
```

Fig. 13. Pseudo code of writing on playout buffer

2) APBC 읽기 동작

최종적으로 수신단의 음성 코덱은 APBC 읽기 동작을 통해 재생 버퍼 상에 프레임 단위로 기록된 음성 데이터를 가져와서 아날로그 음성 신호로 복원한다. APBC 읽기 동작을 수행하는 프로세서는 APBC 기록 과정과는 독립적으로 수행되며, 음성 세션이 시작되어 첫 번째 패킷이 수신되면 그 첫 번째 패킷의 버퍼링 지연만큼 대기한 후에 구동이 시작된다. APBC 읽기 동작은 단순히 재생 버퍼로부터 순차적으로 프레임 데이터를 읽어 와서 재생하면 되는 것으로, 단 상태 표시자가 'PACKET LOSS'를 나타내는 경우에는 현재 읽으려는 프레임 이전으로 가장 근접하게 저장되어 있는 프레임 데이터를 검색하여 반복 재생한다. Fig. 14는 이러한 APBC 읽기 과정을 나타낸 것으로 APBC 기록 후에 버퍼링 지연만큼 프레임 데이터가 쌓이면 읽기 동작을 수행함을 표현하고 있다.

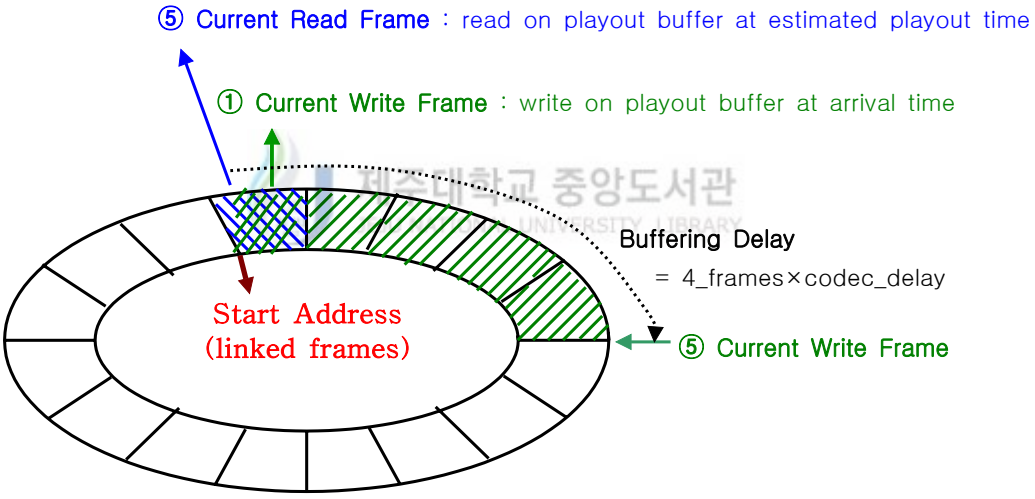


Fig. 14. APBC read/write operation

4. 모의실험

APBC의 기능을 검증하기 위한 모의실험은 APBC 블록을 C 언어로 구현한 후

에 PC 상에서 트래이스 데이터를 이용하며 오프라인으로 수행하였다. 즉 트래이스 데이터를 구하고, 거기에서 각 패킷의 생성 타임스탬프와 수신 타임스탬프, 순서 번호 정보를 취하여 APBC 블록으로 입력한다. 이에 따라 APBC 블록은 동작하여 그 결과 값을 파일에 기록한다. 이는 Fig. 15와 같다.

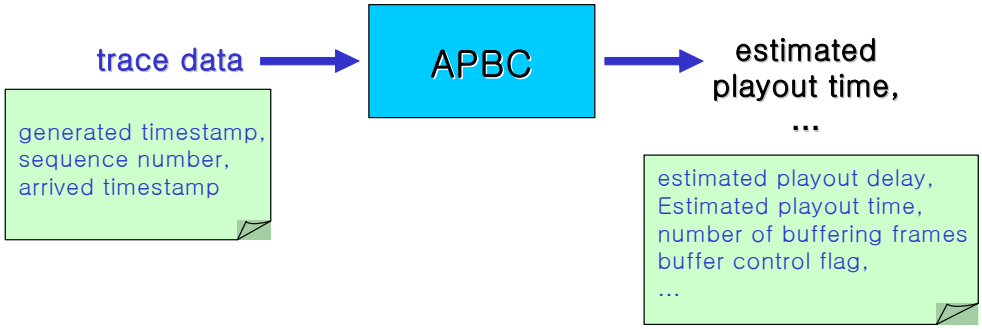


Fig. 15. Simulation of APBC

Fig. 16은 입력 트레이스 데이터를 나타낸 것이고 Fig. 17은 그에 따른 APBC 블록의 동작 과정을 수행하면서 부분 결과 값들을 모니터링 한 것이다. Fig. 18은 APBC 읽기 동작을 수행하여 재생 버퍼에 저장된 프레임 데이터를 나타낸 것이다.

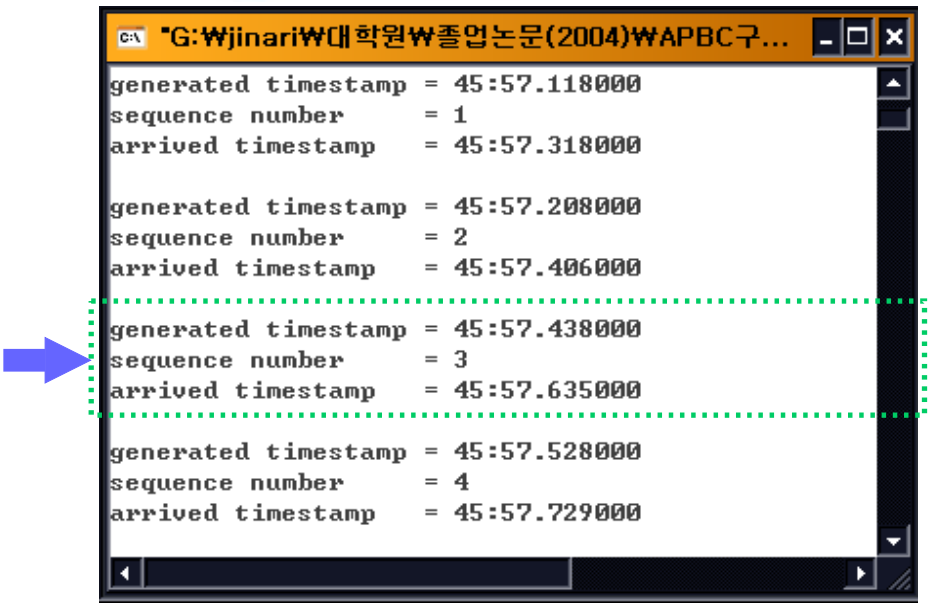


Fig. 16. Input trace data for simulation

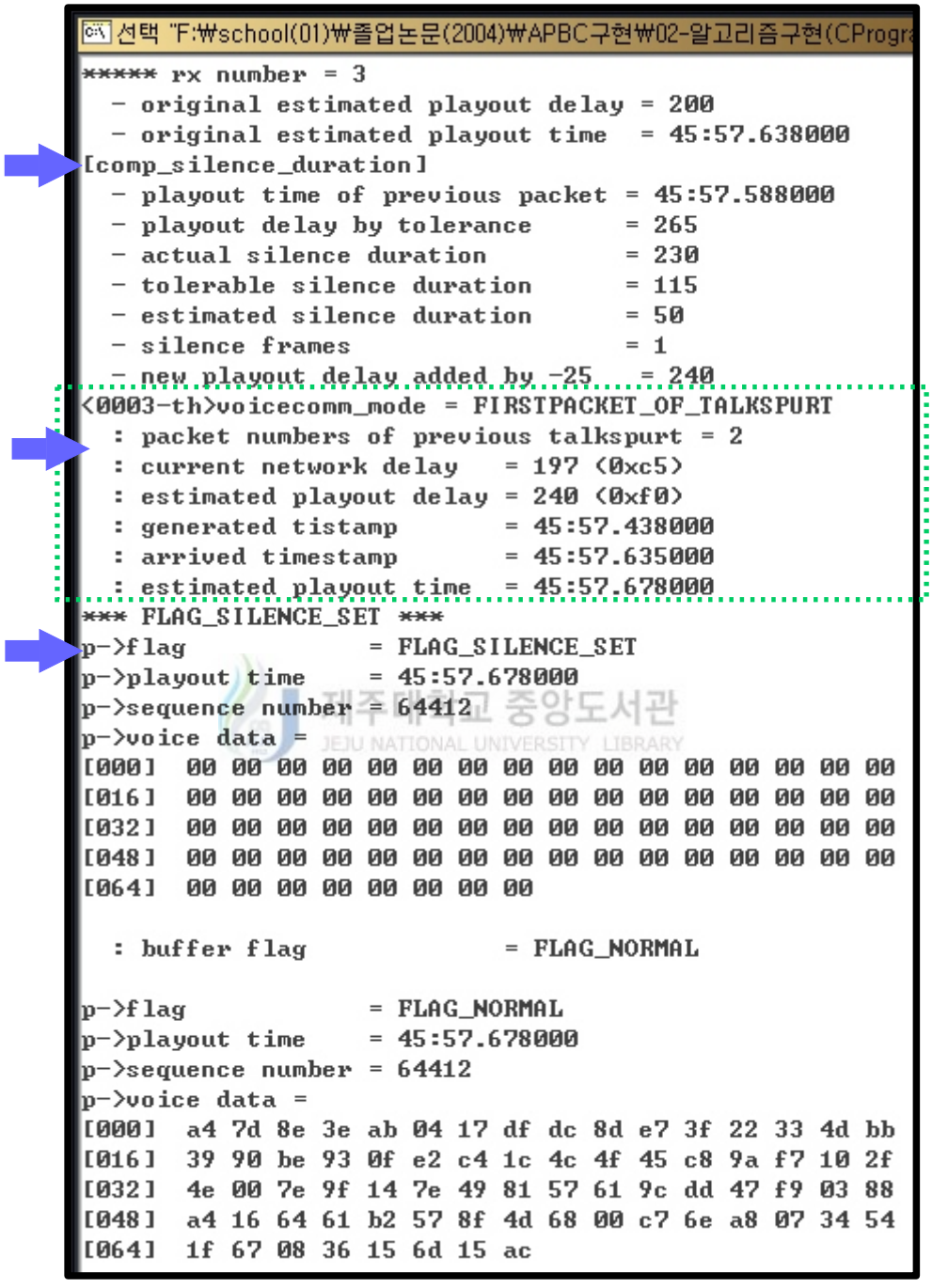


Fig. 17. Monitoring of APBC operating simulation

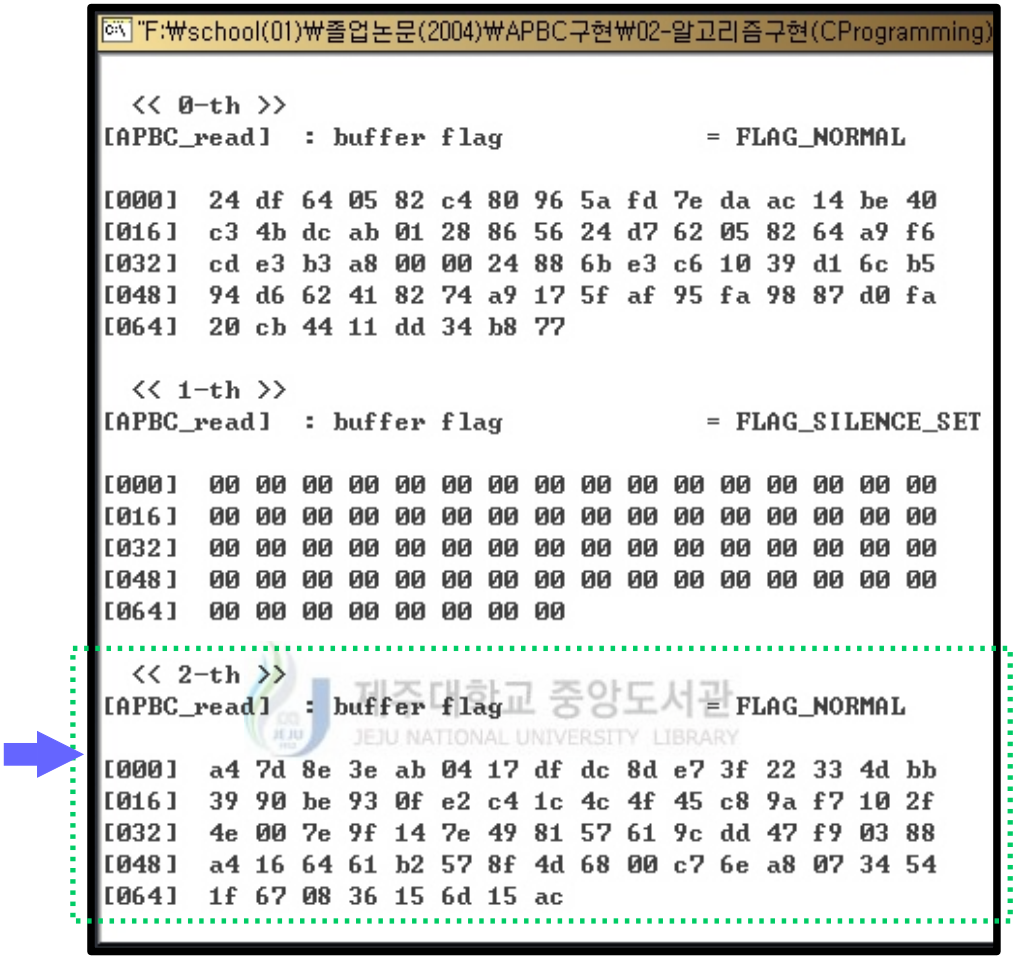


Fig. 18. Frame data obtained by APBC reading operation

다음의 표 1과 표 2는 5번의 패킷 수신에 대한 트래이스 데이터와 그에 대한 APBC의 수행 결과를 나타낸 것이다.

Table. 1. Input trace data

talkspurt number	packet number	sequence number	generated timestamp [hour:minute:μsec]
1	(session start) 1	64410	45:57:118000
1	2	64411	45:57:208000
2	1	64412	45:57:438000
2	2	64413	45:57:528000
2	3	64414	45:57:618000

Table 2. Simulation result

talkspurt number	packet number	seq number	arrived timestamp [hour:minute:μsec]	estimated playout delay [m sec]	buffer frame number	estimated playout time [hour:minute:μsec]
1	1	64410	45:57:318000	380	2	45:57:498000
1	2	64411	45:57:406000	380	1	45:57:588000
2	1	64412	45:57:635000	240	1	45:57:678000
2	2	64413	45:57:729000	240	1	45:57:768000
2	3	64414	45:57:818000	240	1	45:57:858000

실제 음성 통신에서의 수신 패킷들은 RTP 헤더를 통해 순서번호와 생성 타임스탬프 정보만을 제공한다. 표 1에서 음성 구간의 수와 패킷 번호는 APBC가 정확히 동작하는지를 확인하기 위해 송신단에서 트래이스한 정보이다. 표 2는 APBC가 다음과 같이 동작함을 확인한 결과 값이다. APBC는 RTP 헤더의 순서번호에 맞게 패킷을 순차적으로 정렬하고, 입력 트래이스 데이터(표 1)와 동일하게 두 개의 음성 구간을 구분하며, 각 음성 구간에 대한 재생 지연을 380 [m sec]와 240 [m sec]로 추정하였다. 또한 추정된 재생 지연을 정확한 재생 버퍼 프레임 수로 계산하였음이 확인된다.

Ⅳ. VoIP 시스템 구현 및 APBC 적용

VoIP 기술을 이용한 대표적인 응용 시스템은 인터넷 전화이다. 이는 비용이 저렴하고 인터넷 망을 이용하는 다양한 서비스와 연계되어 사용될 수 있다는 장점에 기인하여 다양한 형태로 개발되고 있다. 기기 구조적인 측면에서 인터넷 전화를 분류하면 PC 기반형과 독립형(stand alone)으로 나눌 수 있다. PC 기반형은 PC에 내장된 사운드 카드나 USB와 같은 외부 인터페이스에 새로운 음성 코덱 장치를 연결하여 이를 PC 응용 프로그램이 제어하는 형태이다. 독립형은 VoIP 통화 기능을 제공하는 하나의 내장형 시스템으로 마이크로프로세서와 음성 코덱, 이더넷 컨트롤러 등을 통합 설계한 하드웨어 시스템 또는 이러한 블록들을 통합한 SoC (system on chip) 플랫폼의 하드웨어 시스템을 펌웨어가 제어하는 형태이다.

본 장에서는 먼저 실제 VoIP 응용 시스템을 구현한다. VoIP 응용 시스템으로는 휴대용에 적합한 고성능 저전력 마이크로프로세서를 기반으로 하는 내장형 시스템을 구현한다. 이를 위해 마이크로프로세서와 음성 코덱, 이더넷 컨트롤러를 내장한 보드를 제작하고 운영체제와 인터넷 전화 응용 프로그램을 통합하여 펌웨어를 구현한다. 다음으로는 Ⅲ장에서 설계한 APBC 기법을 구현 시스템에 적용하여 통화 품질 개선 효과를 검증한다.

1. 내장형 VoIP 시스템 구현

1) 기반 시스템 설계

내장형 시스템의 주요 성능을 결정하는 마이크로프로세서는 핸드헬드(handheld) 컴퓨팅 응용에 적합한 고성능 저전력 프로세서인 XScale 계열을 선택하였다. XScale은 기본적으로 32비트 ARM 구조를 갖으며 다양한 표준 인터페이스를 제공함으로써 여러 가지 응용이 가능하다. IP 망과의 통신 인터페이스는 내장형 플

랫폼에서 많이 사용하는 이더넷(ethernet) 컨트롤러로써 성능이 안정적이고 저가인 CS8900을 선택한다. 메모리는 부트로더(boot loader)를 플래시 메모리의 일정 영역에 기록하도록 하여 EEPROM을 사용하지 않고 플래시 메모리와 DRAM을 장착한다. 음성 코덱은 DSP(digital signal processor) 기반의 G.723.1을 이용한다.

운영체제는 이더넷 컨트롤러나 음성 코덱과 같은 외부 연결 장치의 드라이버 개발이 쉽고 로우 레벨 패킷 캡쳐를 위해 소스 코드를 수정할 수 있는 내장형 리눅스를 선택한다. 또한 ITSP(internet telephony service provider)와의 연동을 위한 시그널링 프로토콜(signalling protocol)은 현재까지 상용화 면에서 우수한 H.323으로 구현한다. 표 3은 이상의 시스템 설계에 대한 주요 파라미터를 나타낸 것이다.

Table 3. Design parameters of embedded VoIP system

System Parameters	Format	Performance
Microprocessor	Intel XScale™ (ARM, 32bit)	400Mhz
Network Interface	IEEE 802.3 Ethernet	10BASE-T
Speech Codec	G.723.1 (OakDSPCore™)	50MIPS fixed point DSP core
Memory	SDRAM	64Mbyte
	Flash memory	16Mbyte
Operating System	embedded Linux	kernel version - 2.4.18
VoIP Signalling Protocol	H.323	version 2

2) 하드웨어 구현

Fig. 19는 설계한 내장형 VoIP 시스템의 하드웨어 구성도를 나타낸다. 이는 크게 음성 코덱 모듈과 외부 메모리 모듈, 사용자 인터페이스 모듈, 그리고 IP 망과의 통신을 위한 이더넷 모듈로 구분된다. 사용자 인터페이스는 문자형 LCD(liquid crystal display)와 PC의 COM 포트에 연결되는 콘솔 포트, 키패드(keypad)로 구성한다. 각 모듈에 대한 시스템의 메모리 할당은 Fig. 20과 같이 하였다.

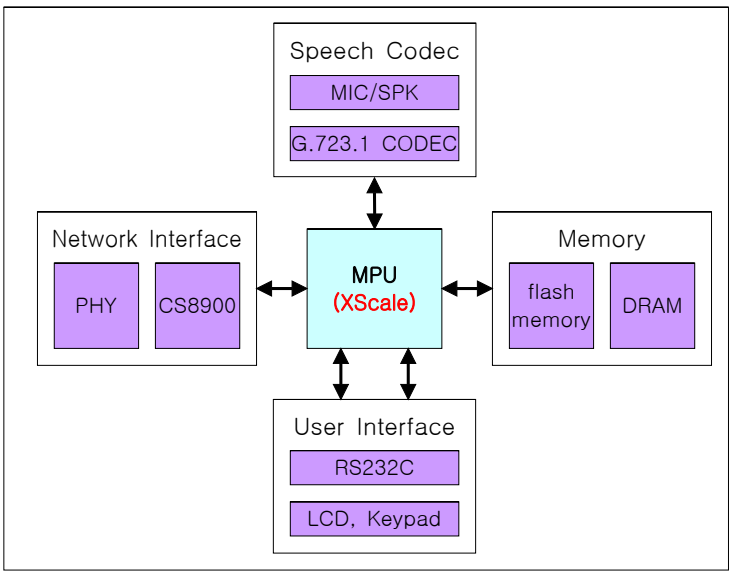


Fig. 19. Hardware structure of embedded VoIP system designed

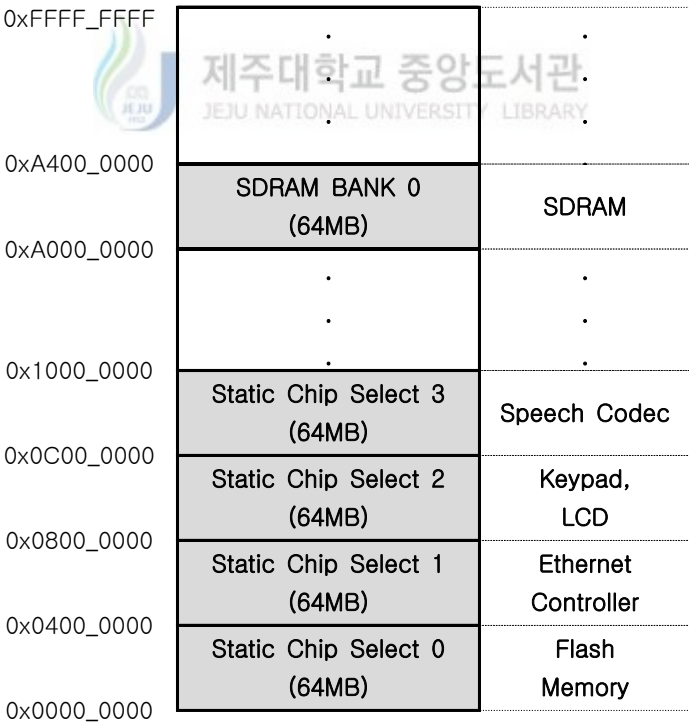


Fig. 20. System memory map

Fig. 21은 마이크로프로세서와 음성 코덱 간의 연결 구성도를 나타내고 있다. 음성 코덱은 마이크로프로세서의 chip-select 3 영역에 연결되며 하나의 IRQ(interru-pt request)를 할당한다. 마이크로프로세서가 코덱에 대해 제어 명령을 내리거나 제어 상태를 점검하고, 또한 재생할 데이터를 넘겨주거나 녹음된 데이터를 가져오기 위해 4비트의 주소 라인과 8비트의 데이터 라인을 연결한다. IRQ 신호는 음성 코덱이 주기적으로 음성 신호 처리 즉, 녹음과 재생을 완료할 때마다 발생된다. 이에 따라 마이크로프로세서에서는 해당 IRQ로부터 들어오는 신호를 감지하여 녹음 데이터는 가져오고 재생 데이터는 넘겨준다.

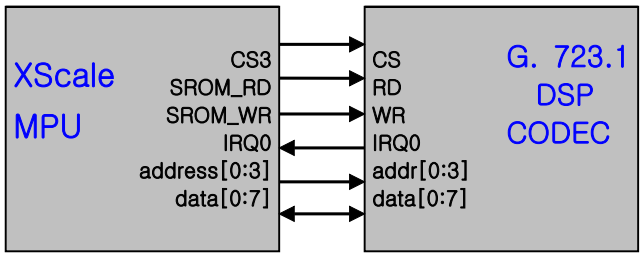


Fig. 21. Connection between MPU and speech codec

Fig. 22는 앞서 설계한 구성도에 의해 제작된 PCB(printed circuit board)를 보여주고 있다.

Fig. 22. Implemented PCB

3) 펌웨어 구현

구현 시스템의 전체 펌웨어 구조는 Fig. 23에 나타내었으며 이는 내장형 시스템의 일반적인 펌웨어 구조와 같이 부트로더와 운영체제 그리고 응용 프로그램으로 대분류된다.

마이크로프로세서 구조에 맞게 포팅된 내장형 리눅스 운영체제에 기반하여 음성 코덱 장치 드라이버와 VoIP 프로토콜 스택 등의 응용 프로그램을 구현한다. 이를 하나의 실행파일로 만들어 플래시 메모리 영역에 기록해두면, 시스템이 초기화될 때마다 부트 로더가 펌웨어를 디램 영역으로 복사하여 구동시킨다.

Fig. 23. Firmware structure of the implemented system

Fig. 24는 전체 펌웨어에서 VoIP 부분을 나타내고 있다. 음성 코덱 장치 드라이버를 이용하여 주기적인 음성 프레임 데이터를 생성하고 RTP/UDP/IP에서는 생성된 데이터를 음성 패킷화하여 전송한다. 반대로 음성 패킷을 수신하여 RTP/UDP/IP 블록을 통해 음성 프레임 데이터를 분석하여 음성 코덱 장치 드라이버에 넘겨줌으로써 음성 프레임 데이터를 재생한다. H.323 프로토콜 스택은 음성 통신 이전에 대화자 간에 음성 세션을 연결하는 기능을 한다.

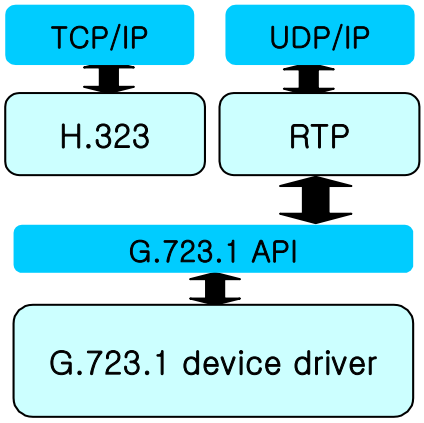


Fig. 24. Structure of firmware associated VoIP communication

2. 통화 기능 검증

구현 시스템의 기능 검증은 실제 IP 망을 통한 음성 통화 시험으로써 수행한다. 이를 통해 구현한 VoIP 프로토콜 스택의 호 설정 기능과 RTP 기반의 음성 패킷 송수신 기능 및 코덱을 이용한 음성 신호의 압축과 복원 기능 등을 검증할 수 있다. 단계적으로는 단방향 실시간 음성 패킷 전송 시험을 거친 후 양방향 음성 통화 시험을 수행한다.

1) 단방향 음성 패킷 전송 시험

Fig. 25는 본 연구에서 수행한 단방향 실시간 음성 패킷 전송 시험 구성도를 나타낸 것이다. 구현 시스템에서는 주기적으로 음성 신호를 부호화하여 음성 패킷을 송신하고, 원격지의 PC에서는 네트워크 분석 프로그램을 이용하여 수신되는 음성 패킷의 도달 상태를 확인하도록 한다. 이 때 구현 시스템을 제주에 두고, 원격지 PC는 서울에 둔다.

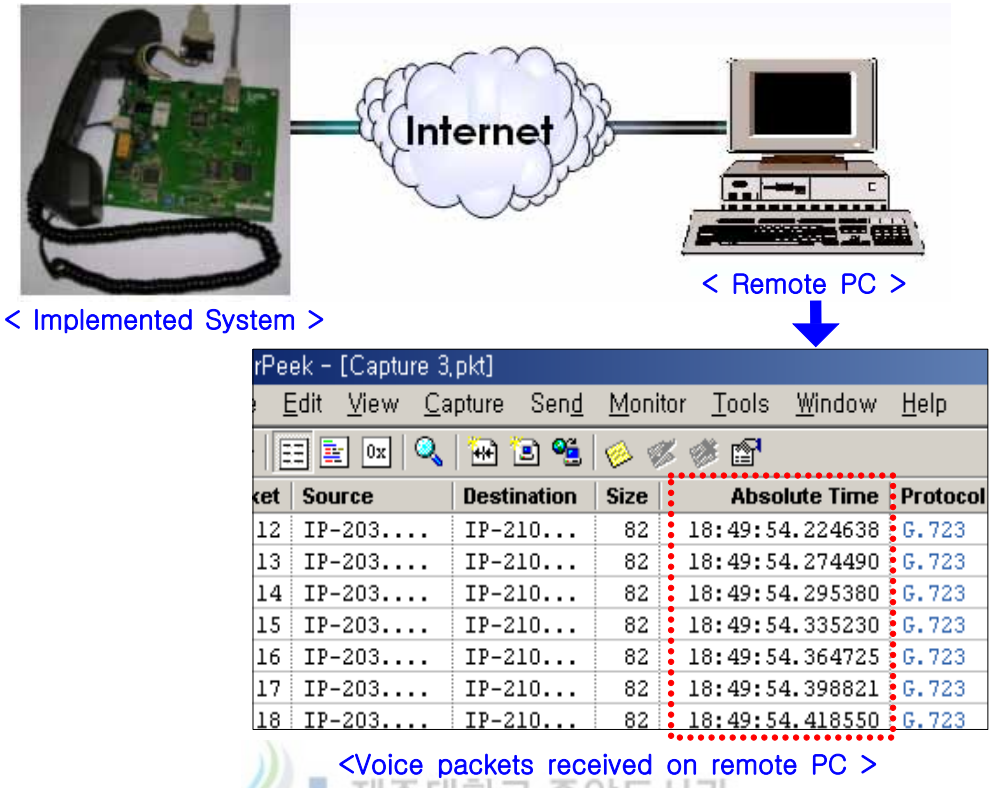


Fig. 25. Test of voice packet transfer in realtime by one way

Fig. 25에서 원격지 PC에서의 음성 패킷 수신 시간 측정 부분은 실시간으로 전송된 음성 패킷의 수신 시간을 보여주고 있다. 100 패킷에 대하여 음성 패킷들의 수신 시간 간격을 측정한 결과 약 40 [m sec]로 정상적으로 전송이 되고 있음을 확인하였다.

2) 음성 통화 시험

VoIP 시스템의 통화 시험은 IP 망에 연결된 VoIP 시스템 간의 통화와 VoIP 게이트웨이를 이용한 IP 망에 연결된 VoIP 시스템과 PSTN(public switched tele-phone network) 망에 연결된 일반 전화 간의 통화로 이루어질 수 있다. 일반적으로 첫 번째의 경우를 IP Phone-to-IP Phone으로 표기하고 두 번째의 경우는 IP Phone-to-Phone이라고 표기하며, 두 경우에 대해 각각 Fig. 26, Fig. 27과 같이 구성하여 시험하도록 한다.

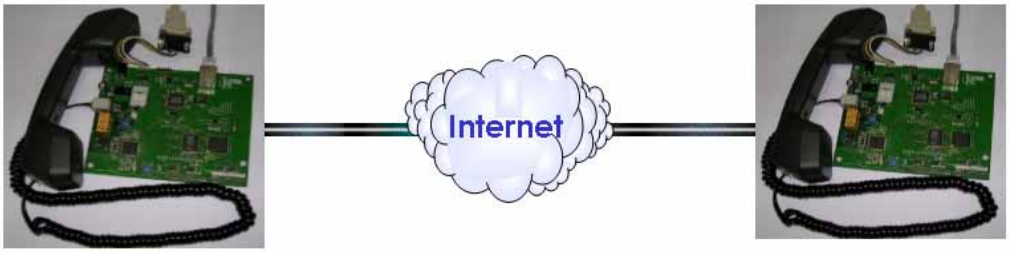


Fig. 26. Speech test 1: IP Phone-to-IP Phone

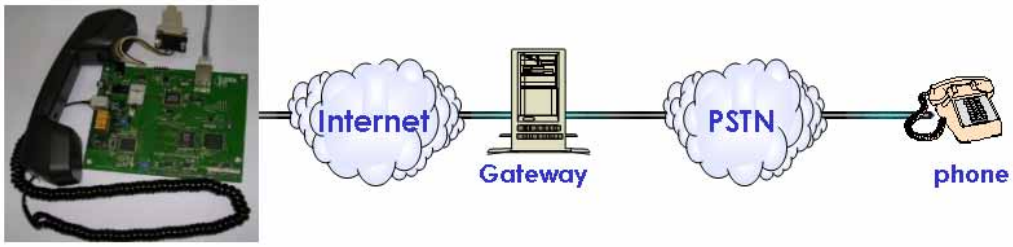


Fig. 27. Speech test 2: IP Phone-to Phone

IP Phone-to-Phone 테스트에서 VoIP 게이트웨이는 H.323 프로토콜 지원 장비를 사용하였으며 두 시험에서 음성 통화가 성공적으로 이루어짐을 확인할 수 있었다.

3. APBC 성능 검증

APBC는 C 언어로 구현된 하나의 알고리즘으로 구현 시스템에 적용하기 위해서는 APBC 알고리즘의 입·출력을 정의하여 시스템에서 제공해야 하는 정보와 결과값에 따른 시스템의 처리 과정을 연결해야 한다. 이는 구현 시스템에 포팅된 내장형 리눅스 운영체제의 자원을 이용하여 구현한다.

Fig. 28은 구현 시스템의 펌웨어 내에서 APBC 수신단의 입·출력을 연결한 형

태를 나타낸 것이다. 펌웨어는 운영체제의 UDP 소켓으로부터 음성 패킷을 수신하여 APBC 블록으로 패킷 데이터와 패킷 수신 시간을 입력하고, 이에 따라 APBC 블록은 기능을 수행하여 프레임 데이터를 재생 버퍼에 저장한다. 또한 구현 시스템의 음성 코덱은 재생 버퍼에 저장된 프레임 데이터를 가져가서 재생하도록 구현하였다.

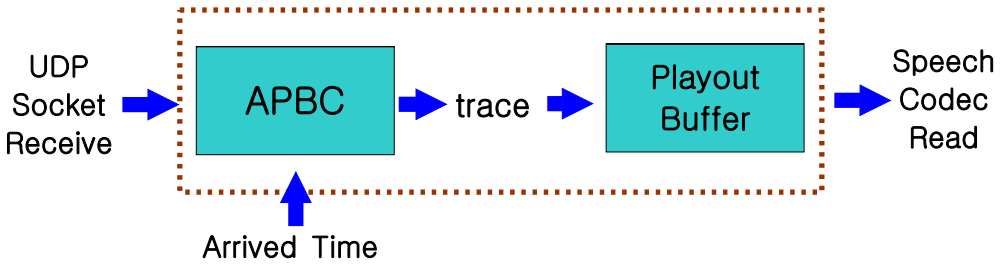


Fig. 28. Connection of APBC block with implemented system

Fig. 29는 구현 시스템에서의 APBC 수행 속도를 측정한 방법을 나타낸 것이며 수행 속도는 257 [μ sec]로 측정되었다. 이는 음성 코덱의 단위 음성 신호 처리 시간보다 월등히 작은 값으로 이는 APBC가 실시간 처리에 적합하게 수행됨을 확인할 수 있다.


```
(1) WHILE(there is a packet in a trace file)
(2) Fetch a packet;
(3) First checkpoint;
(4) Packet processing of the algorithm;
(5) Microsecond checkpoint;
```

Fig. 29. Measurement of APBC processing time

표 4는 구현한 내장형 VoIP 시스템에 고정 재생 지연 방식을 사용하는 Open H.323을 적용하였을 때와 APBC 기법을 적용하였을 때의 통화 품질을 비교한 것이다. 구현 시스템을 트래픽이 많은 IP 망 환경에 둔 상태에서 원격지의 VoIP 게

이트웨이를 통하여 일반 전화와 통화 하는 경우에, 구현 시스템을 사용하는 사람 측에서 MOS를 측정하였다. 대화 내용은 동일하며 약 30초에 걸쳐 통화를 수행하였다. 총 10인의 대화 상대에 수행하였으며 표 4는 총 상대에 대한 평균치를 나타낸 것이다.

Table 4. The Performance comparison of APBC with Open H.323

	Open H.323	APBC
Speech Quality [MOS]	2.7	3.2

표 4에서 고정 재생 지연 방식을 사용하는 Open H.323을 적용하였을 때의 통화 품질은 2.7이고 제안 기법을 적용하였을 때의 통화 품질은 3.2로 측정이 되어 18%의 개선 효과를 가져왔다.

Ⅴ. 결론

본 논문은 VoIP 시스템의 통화 품질 개선을 위해 적응 재생 알고리즘을 도입하여 지터를 보상하고, 패킷 손실 은닉 및 패킷 순서 정렬을 수행하는 수신단 적응 재생 버퍼 제어 기법을 제안하였다. 또한 제안 기법의 효용성을 입증하기 위해 VoIP 응용 시스템을 구현하고 제안 기법을 적용하여 알고리즘의 수행 속도와 통화 품질 개선 효과를 검증하였다.

이를 수행하기 위해, 먼저 IP 망에서 발생하는 통화 품질 저해 요소와 상관관계를 살피고 현재까지 제안된 적응 재생 알고리즘들을 검토하였다. 이를 토대로 α-adaptive 알고리즘과 지터 제어 절차를 도입하기 위해 적응 재생 버퍼 구조와 제어 방법을 설계하였다. 또한 RTP 헤더를 분석하여 패킷 순서 뒤바뀜 현상과 패킷 손실 발생 여부를 판단하여, 이에 대한 보상이 재생 버퍼 상에서 이루어지도록 설계하였다. 제안 기법을 C 언어로 구현하여 PC 상에서 모의 실험한 결과, 입력 트래이스 데이터에 대하여 묵음 구간과 음성 구간을 정확히 판단하고 음성 구간에 대한 재생 지연을 적응적으로 추정하며 이에 따라 수신 패킷 데이터를 재생 버퍼 상의 정확한 위치에 씀을 확인하였다. 또한 읽기 동작을 통해 추정된 재생 지연을 유지하면서 재생 버퍼 상의 패킷 데이터를 얻음을 확인하였다. 다음으로는 XScale 구조의 마이크로프로세서와 DSP 기반의 G.723.1 음성 코덱을 내장한 보드를 제작한 후, 내장형 리눅스를 탑재하여 H.323 기반의 VoIP 펌웨어를 구현함으로써 내장형 VoIP 시스템을 제작하였다. 시스템의 기능 검증은 IP Phone-to-IP Phone, IP Phone-to-Phone 통화 시험을 통해 수행하였으며 두 경우에 대해 성공적으로 통화가 이루어졌다. 최종적으로 구현 시스템에 제안 기법을 적용하여 성능을 측정하였다. 그 결과, 제안 기법의 알고리즘 수행 속도는 257 [μ sec]로 측정되었고, 이는 음성 코덱이 하나의 패킷 데이터를 처리하는 시간에 비해 매우 작은 값으로, 본 논문에서와 같이 G.723.1 음성 코덱을 사용하는 경우에 코덱이 처리하는 시간에 비한 알고리즘의 수행속도는 0.8%에 해당한다. 따라서 제안 기법은 실시간으로 수신되는 패킷들을 처리하기에 적합하다. 또한 통화 품질 만족도를 MOS로 측정한

결과, 동일한 환경에서 고정 재생 지연 방식을 사용하는 알고리즘에 비해 18% 만큼 개선되었다.

본 논문에서는 VoIP 통화 품질 개선을 위한 수신단 패킷 처리 기법을 제안하고 VoIP 응용 시스템 상에서의 적용 가능성과 통화 품질 개선 효과를 검증함으로써, VoIP 응용 시스템이 보다 안정적인 통화 품질을 제공하기 위한 하나의 방안으로 제안 기법이 적합할 것으로 사료된다. 추후에는 지속적으로 좀 더 다양한 망 환경에서의 성능 개선 효과에 대한 검증이 보완될 것이며 MOS 이외의 다양한 성능 측정 방법을 이용한 검증이 수행되어야 하겠다.

참고 문헌

- DeLeon P. and Sreenan C. J., 1999, An Adaptive Predictor For Media Playout Buffering, *IEEE ICASSP99*, 6 3097~3100
- Johnson J. T., IP Telephony: State of the Market, *NGN2003*, pp. 50
- Kansal A. and Karandikar A., 2001, Adaptive Delay Estimation for Low Jitter Audio over Internet, *IEEE Global Telecommunications Conference*, 4 2591~2595.
- 김도영, 김영선, 2004, 인터넷 전화 기술 현황 및 전망, 한국통신학회지 21(4) 19~28.
- 이인화, 박종계, 2004, 인터넷전화 도입을 위한 기술 및 시장의 주요 이슈, 한국통신학회지 21(4) 29~38.
- Moon S. B., Kurose J. F., and Towsley D. F., 1998, Packet audio playout delay adjustment: Performance bounds and algorithms, *ACM/Springer Multimedia Systems*, 6 17~28.
- Pinto J. and Christensen K. J., 1999, An Algorithm for Playout of Packet Voice based on Adaptive Adjustment of Talkspurt Silence Periods, *LCN*, 22 4~231.
- Ramjee R., Kurose J. F., Towsley D. F., Schulzrinne H., 1994, Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks, *IEEE INFOCOMM,* 680~686
- Schulzrinne H., 1992, Voice Communication Across the Internet: A Network Voice Terminal, Technical Report University of Massachusetts 34 pp.
- 유승화, 2002, 인터넷 전화, 전자신문사, pp.222~236

URL :

- http://www.voiptroubleshooter.com/indepth/jittersources.html

- http://www.cs.columbia.edu/~hgs/rtp/news.html
- http://www.ietf.org/rfc/rfc1889.txt

감사의 글

뒤늦게 시작하여 우여곡절 속에 논문을 완성해가는 동안, 생각치도 못했던 많은 것들을 배우게 됩니다. 그 동안의 고민과 노력, 지면에서 지워진 많은 생각과 글들, 곁에서 도와준 사람들의 따뜻한 격려야말로 가장 값진 결실이 아닌가 합니다.

학부, 직장, 대학원을 거쳐 가는 동안 항상 옆에서 격려해주시고 다양한 기회를 만들어주신 임재윤 교수님께 가장 먼저 감사드립니다. 기대이상으로 해내지 못한 데에 대한 송구스러움과 지금까지 받은 사랑에 대한 깊은 감사를 전합니다. 교수님 가르침 잊지 못할 것입니다. 바쁘신 일정 속에서도 부족한 논문을 보완하는 데에 마지막까지 아낌없는 도움을 주신 김홍수 교수님께 감사드리고, 대학원 과정 동안 실제 많은 가르침을 주시고 논문 마무리까지 힘이 되어주신 좌정우 교수님께 감사드립니다. 학부 때부터 항상 따뜻한 얼굴로 맞아주시고 가르침 주신 이용학 교수님과 문건 교수님, 열정적인 강의와 가르침으로 도움주신 양두영 교수님과 강진식 교수님께도 감사드립니다.

올바른 생각을 전해주시고 후배들을 이끌어주심에 아낌없는 부식선배님과, 잠시나마 같이 연구하게 되어 영광인 성욱선배님, 대학원 생활 동안 힘이 되어 주신 봉수선배님께 감사드립니다. 또한 힘든 박사논문 쓰시면서도 후배들 논문까지 봐주시느라 고생 많으셨던 권익선배님, 정말 감사했습니다. 항상 웃는 얼굴로 맞아주고 도움 주신 조교선생님 윤희와 진호오빠에게도 고마움을 전합니다. 대학원 입학 때부터 졸업하기까지 관심과 격려 보내준 진숙, 수미 그리고 고마운 철우오빠, 영길오빠, 광식오빠 감사합니다. 같은 연구실 선배로써 너무나도 많은 도움 줘서 말로 다 못하는 창윤오빠와 닮고 싶은 영애언니, 어려운 문제가 생길 때마다 아낌없는 도움 주는 재필오빠 감사합니다. 대학원에서 같이 연구할 때 늘 힘이 되어주고 사수가 되어준 고마운 영배오빠와, 내 마음을 나만큼이나 잘 알아주며 힘들때 마다 묵묵히 도와준 현미에게 고마움을 전합니다. 그리고 같은 연구실에서 동고동락하면서 실수할 때마다 넓은 마음으로 이해하고 감싸준 군선오빠에게도 깊은 감사를 전합니다. 다음 학기에 졸업하게 될 상보오빠, 좋은 논문 쓰리라 믿으며 힘내시구요, 대학원 생활 내내 힘이 되어준 마음 잘 통하는 친구 훈철아, 고맙다. 그리고 성실한 우리 연구실 후배들 영미, 현숙, 경준, 용범, 윤철, 윤길 모두 고맙습니다.

논문 쓰는데 도움 주신 (주)인터에프씨 직원 모두에게도 감사드립니다. 특히 나의 정신적인 인도자이면서 실제 논문연구에서도 가장 많은 도움을 준 영주에게 깊은 감사의 마음을 전하고 아프지 말고 건강하길 바랍니다. 운영체제 부분을 맡아서 해준 믿음직한 정현오빠에게도 진심으로 감사드립니다. 그 외에도 단짝친구 인경이와 성실한 성민오빠, 마음 따뜻한 재오오빠, 다른 무대에서 활약하고 있는 행석오빠와 현주 모두 어려울 때 힘이 되어 주셔서 감사합니다.

마지막으로 저의 뿌리인 사랑하는 부모님의 변함없는 믿음과 사랑에 감사드리며, 뒤에서 후원해주시는 언니와 형부, 귀여운 조카 재혁이, 그리고 새로운 가정을 이룬 오빠와 새언니의 깊고도 따뜻한 사랑에 감사드립니다.